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 [6], later refined in [5]. Using the techniques described in [17, 24] we can define the rewriting rules over the distributions.

If we denote by $[(p_1, t_1), \ldots, (p_n, t_n)]$ 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.

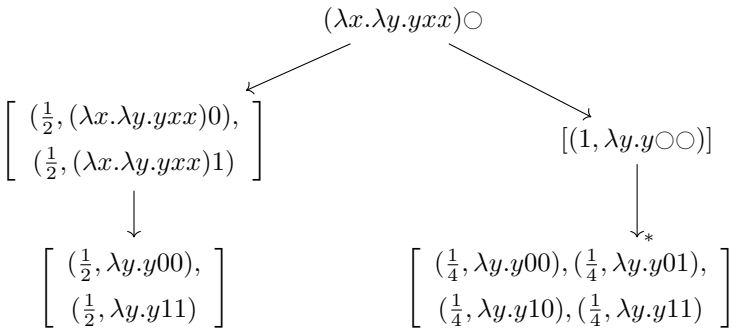


Figure 1.1: *Counterexample of confluence*

$$(\lambda x.t)r \to_1 t[r/x] \qquad \mathsf{if}\ 1\ \mathsf{then}\ t\ \mathsf{else}\ u \to_1 t \qquad \bigcirc \to_{\frac{1}{2}} 1$$
$$\mathsf{if}\ 0\ \mathsf{then}\ t\ \mathsf{else}\ u \to_1 u \qquad \bigcirc \to_{\frac{1}{2}} 0$$

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

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

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

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

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

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

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

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 $t \to_{p_1} r_1 \ldots \to_{p_n} r_n$ we may write $t \to_1^* r_n$, where $p = \prod_{i=1}^{n} p_i$.

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

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

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

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

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

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

coin toss and it takes at most one step for each operator. Hence, using the rewriting over probabilistic distributions techniques from [17], 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 [17] 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.

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 [12] for a discussion on this choice, and even an alternative combining both approaches).

1.3.2 Internalising the probabilities

Following [16], 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., [1].

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

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

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

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

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

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

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 $(\lambda y.yxx)[\bigcirc/x] = \lambda y.y\bigcirc\bigcirc \to^*_{\frac{1}{4}} \lambda y.y11$.

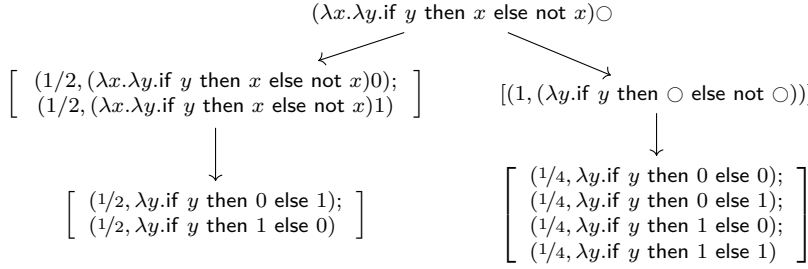
The drawback in this approach is clear, there is a loss in expressivity. Of course, in some cases, this restriction is a desirable quality. For example, in quantum computing it may serve to avoid cloning qubits, a forbidden operation in quantum mechanics.

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



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.

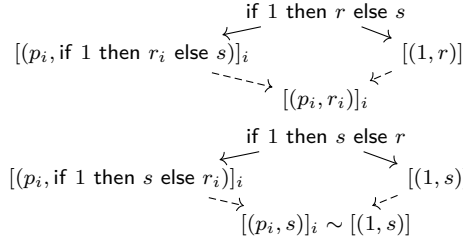
Then, we can prove that the confluence modulo computational equivalence of the sub-affine calculus (Theorem 1.4.4). We need the following two lemmas, which follow by straightforward induction.

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.



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

$$[(p_i, (\lambda x.t_i)r)]_i \longleftarrow (\lambda x.t)r \longrightarrow [(1, t[r/x])]$$
$$\searrow \quad \swarrow$$
$$[(p_i, t_i[r/x])]_i$$

The last critical pair requires a more thorough analysis.

$$[(p_i, (\lambda x.t)r_i)]_i \longleftarrow (\lambda x.t)r \longrightarrow [(1, t[r/x])]$$
$$\searrow$$
$$[(p_i, t[r_i/x])]_i$$

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 chapter we have analyzed the different possibilities to transform a simple probabilistic calculus into a (probabilistically / computationally) confluent calculus. The main result 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 [31] to prove the computational confluence of the quantum lambda calculus λ_ρ [16].

For the following two chapters, we will focus on deterministic quantum lambda calculi which do not perform measurement. This is a deliberate choice, since the inclusion of measurement operations also add the need to account for probabilistic evolutions. We will study different characteristics of these calculi, but model them with insights from this chapter to make a future incorporation of measurement as seamlessly as possible.

Chapter 2

Encoding High-level Quantum Programs as SZX-diagrams

2.1 Introduction

The ZX calculus [33] has been used as intermediary representation language for quantum programs in optimization methods [20, 4, 3] and in the design of error correcting schemes [13].

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 [8], which generalizes the ZX diagrams to work with arbitrary qubit registers using a compact representation. Previous work [7] 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 [23, 15, 32]. Quipper [27] 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 [26, 25] 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.

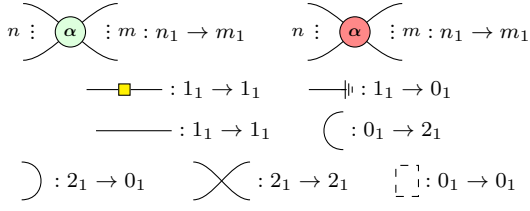
2.2 Background

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 [27], a Haskell-like programming language for describing quantum computations. In this work we use a formalization of the language called Proto-Quipper-D[25] 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.

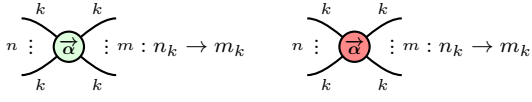
2.2.1 The Scalable ZX-calculus

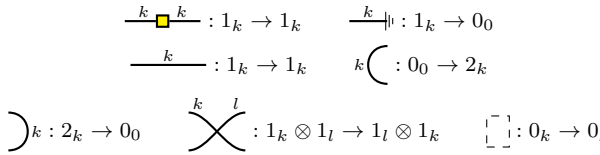
The ZX calculus [33] is a formal graphical language that encodes linear maps between quantum states. Multiple extensions to the calculus have been proposed. We first present the base calculus with the grounded-ZX extension, denoted $\text{ZX}_{\frac{1}{2}}$ [11], to allow us to encode quantum state measurement operations. A $\text{ZX}_{\frac{1}{2}}$ 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.

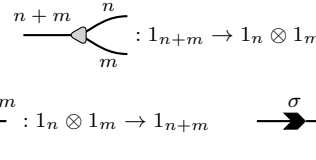
The SZX calculus [8, 7] is a *Scalable* extension to the ZX-calculus that generalizes the primitives to work with arbitrarily sized qubit registers. This facilitates the representation of diagrams with repeated structure in a compact manner. Carette et al. [7] show that the scalable and grounded extensions can be directly composed. We will refer to the resulting $\text{SZX}_{\frac{1}{2}}$-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:



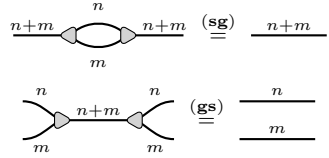


Wires of multiplicity zero are equivalent to the empty mapping. We may omit writing the wire multiplicity if it can be deduced by context.

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*.

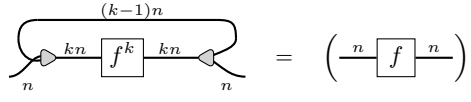


The rewriting rules of the calculus imply that a SZX diagrams can be considered as an open graph where only the topology of its nodes and edges matters. In the translation process we will make repeated use of the following reductions rules to simplify the diagrams:



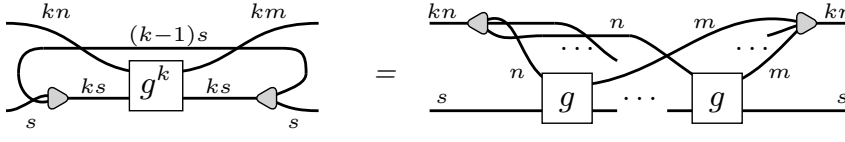
In an analogous manner, we will use a legless gather to terminate wires with cardinality zero. This could be encoded as the zero-multiplicity spider [], which represents the empty mapping. Refer to Appendix .1 for a complete definition of the rewriting rules and the interpretation of the SZX calculus. Cf. [7] for a description of the calculus including the generalized arrow generators.

Carette et al. [7] 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*

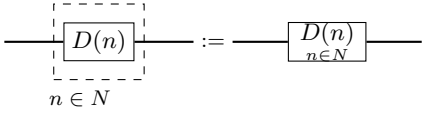


As an example, given a list $N = [n_1, n_2, n_3]$ and a starting accumulator value x_0, this construction would produce the mapping $([n_1, n_2, n_3], x_0) \mapsto ([m_1, m_2, m_3], x_3)$ where $(m_i, x_i) = g(n_i, x_{i-1})$ for $i \in [1, 3]$.

2.2.2 SZX diagram families and list instantiation

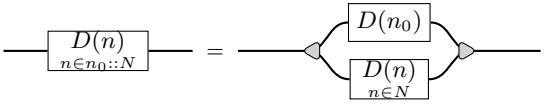
We introduce the definition of a family of SZX diagrams $D : \mathbb{N}^k \to \mathscr{D}$ as a function from k integer *parameters* to SZX diagrams. We require the structure of the diagrams to be the same for all elements in the family, parameters may only alter the wire tags and spider phases. Partial 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. [7], 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.

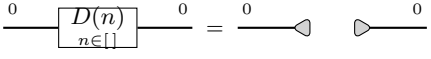


The definition of the list instantiation operator is given recursively on the construction of D in Figure 2.1. On the diagram wires we use $v(N)$ to denote the wire cardinality $\sum_{n\in N} v(n)$, $\overrightarrow{\alpha}(N)$ for the concatenation of phase vectors $\overrightarrow{\alpha}(n_1) :: \cdots :: \overrightarrow{\alpha}(n_m)$, 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:

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

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

Table 2.1: *Type Grammar*

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

Here `U` denotes a set of unitary operations and `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, `new` is used to create a qubit, and `meas` to measure it. $\star$ is the inhabitant of the `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[10].

Types are divided into two kinds; parameter and state types. Both of these can depend on terms of type `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 `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:

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

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

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:

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

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

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

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

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

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

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

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

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

$$\left\lfloor \texttt{let}\ x^P :: y^{\texttt{Vec}\ n\ P} = M\ \texttt{in}\ N \right\rfloor_\Phi =$$
$$|\Phi| \mapsto \lfloor N \rfloor_{x:P, y:\texttt{Vec}\ n\ P, \Phi} (y_1, [y_2, \ldots, y_n], |\Phi|)$$
$$\text{where } [y_1, \ldots, y_n] = \lfloor M \rfloor_\Phi (|\Phi|)$$

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 $A = (n_1 : \texttt{Nat}) \to \cdots \to (n_k : \texttt{Nat}) \to S$, 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.

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

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

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

.

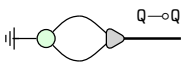
We will omit the brackets in our diagrams for clarity. In a similar manner to the evaluation, we define the translation $[\![M]\!]_{\Phi,\Gamma}$ recursively on the terms as follows:

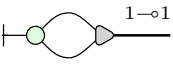
ARREGLAR ESTA PARTE QUE ESTÁ ILEGIBLE

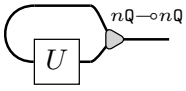
$[\![x]\!]_{\Phi,x:A} = |\Phi| \mapsto$ 

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

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

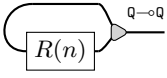
$[\![\texttt{meas}]\!]_{\Phi} = |\Phi| \mapsto$ 

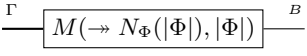
$[\![\texttt{new}]\!]_{\Phi} = |\Phi| \mapsto$ 

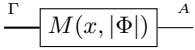
$[\![U]\!]_{\Phi} = |\Phi| \mapsto$ 

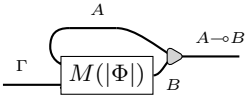
$[\![\star]\!]_{\Phi} = |\Phi| \mapsto$ 

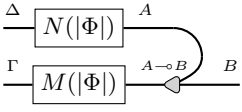
$[\![\texttt{VNil}]\!]_{\Phi} = |\Phi| \mapsto$ 

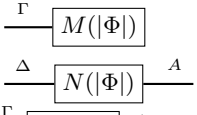
$[\![R]\!]_{\Phi} = n, |\Phi| \mapsto$ 

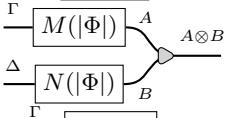
$[\![M\ @N]\!]_{\Phi,\Gamma} = |\Phi| \mapsto$ 

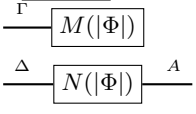
$[\![\lambda' x^A.M]\!]_{\Phi,\Gamma} = x, |\Phi| \mapsto$ 

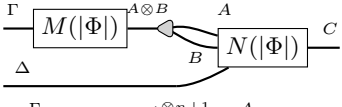
$[\![\lambda x^A.M]\!]_{\Phi,\Gamma} = |\Phi| \mapsto$ 

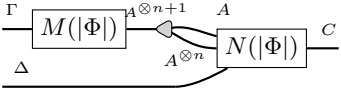
$[\![M\ N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

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

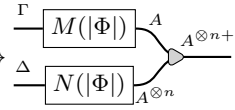
$[\![M \otimes N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

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

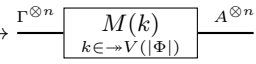
$[\![let\ x^A \otimes y^B = M\ in\ N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

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

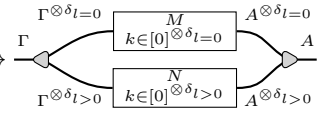
$$[\![M :: N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$$



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



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



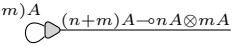
Where δ is the Kronecker delta and $l = \lfloor L \rfloor (|\Phi|)$. Notice that the `new` and `meas` operations share the same translation. Although `new` can be encoded as a simple wire, we keep the additional node to maintain the symmetry with the measurement.

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 chapter.

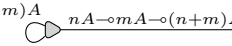
Name	Rz(n)	Rz^{-1}(n)	Rx(n)	Rx^{-1}(n)	H	CNOT
Encoding						

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.

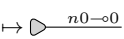
$$[\![\texttt{split}_A]\!]_\Phi = n, m, |\Phi| \mapsto$$



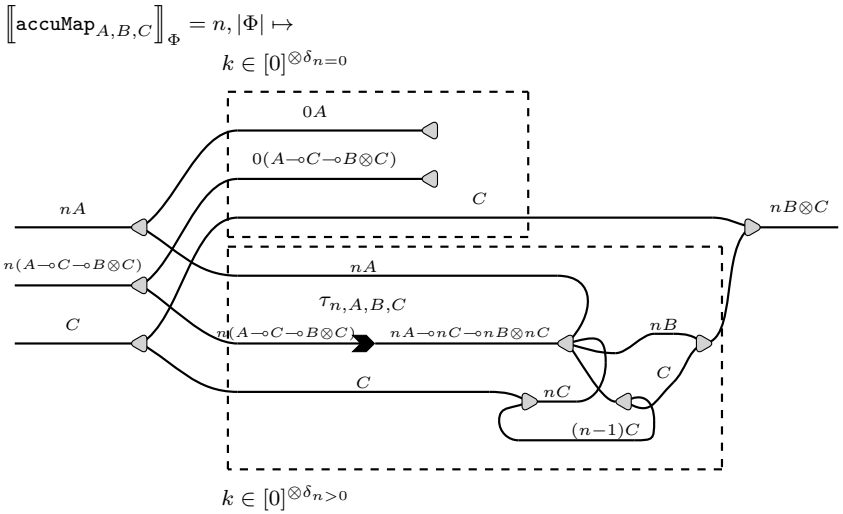
$$[\![\texttt{append}_A]\!]_\Phi = n, m, |\Phi| \mapsto$$



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



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.



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 $(A, C, B, C) \ldots (A, C, B, C)$ into the sequence $(A \ldots A)(C \ldots C)(B \ldots B)(C \ldots C)$. It is defined as follows:

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

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 [2], 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 [29]. The QFT function oper-

ates 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 $n \ldots m$ as a shorthand for `range` $@n$ $@m$.

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

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

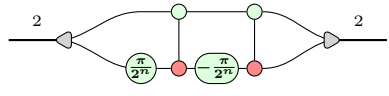
apply_crot: $(n : \texttt{Nat}) \to (k : \texttt{Nat}) \to \texttt{Vec } n\ \texttt{Q} \multimap \texttt{Vec } n\ \texttt{Q}$

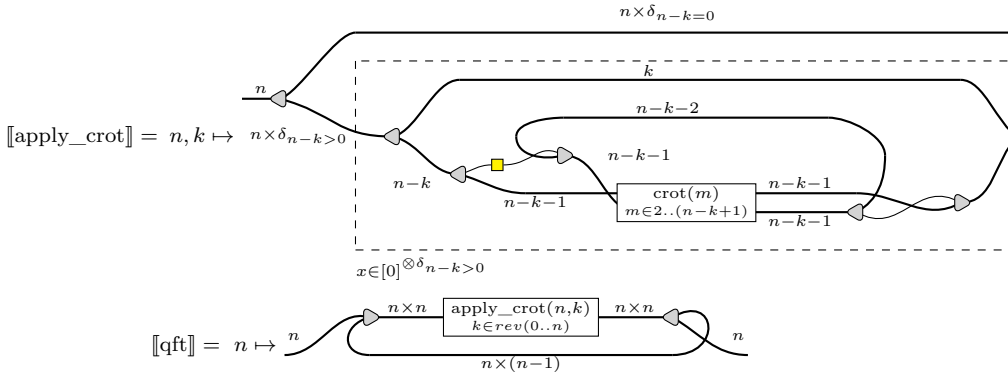
$$\begin{aligned}\text{apply\_crot} :=\ & \lambda' n^{\texttt{Nat}}.\ \lambda' k^{\texttt{Nat}}.\ \lambda qs^{\texttt{Vec } n\ \texttt{Q}}.\\ & \texttt{ifz } (n-k) \texttt{ then } qs \texttt{ else}\\ & \texttt{let } h^{\texttt{Vec } k\ \texttt{Q}} \otimes qs'^{\texttt{ Vec } n-k\ \texttt{Q}} = \texttt{split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

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

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

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.

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



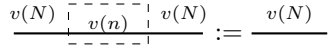
2.6 Conclusion

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.

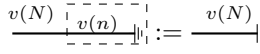
Given $D : \mathbb{N}^{k+1} \to \mathscr{D}$, $N = [n_1, \ldots, n_m] \in \mathbb{N}^m$,

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

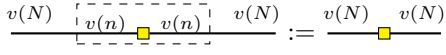


$n \in N$

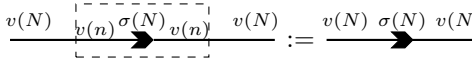
$$((D_2 \circ D_1)(n), n \in N) := (D_2(n), n \in N) \circ (D_1(n), n \in N)$$



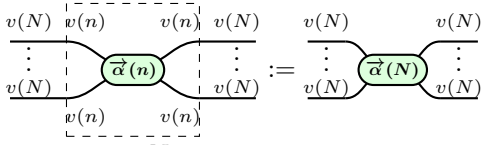
$n \in N$



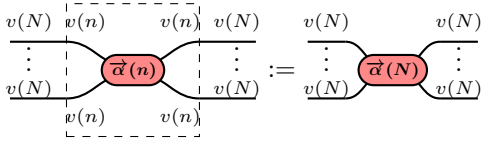
$n \in N$



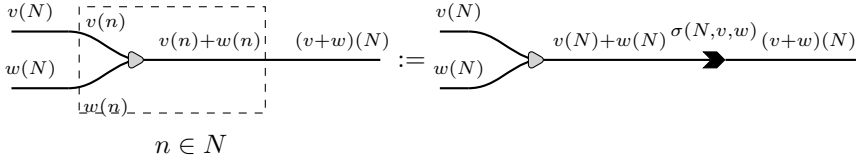
$n \in N$



$n \in N$



$n \in N$



$n \in N$

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

The no-cloning theorem [**?**] and the no-deleting theorem [**?**] state it is impossible to copy or delete an arbitrary qubit. There is, however, a subtlety: although arbitrary qubits cannot be copied or deleted, this is possible for known—and, in the case of deletion, separable—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, to some extent, delete it.

In most quantum programming languages, qubits are defined with respect to a canonical basis—often referred to as the computational basis. In this setting, classical bits correspond to the basis vectors, whereas qubits are unit-norm linear combinations of them. Classical bits can be copied and deleted freely, while such operations on arbitrary qubits are restricted.

In this paper we introduce a quantum λ-calculus within the quantum-control paradigm—by contrast with the classical-control one, where the control flow is classical and cannot be superposed. Our approach is inspired by a line of work on basis-sensitive quantum typing. The earliest system, Lambda-S [**?**], could distinguish whether a qubit was in the computational basis, allowing duplication and erasure only in that case. Later, Lambda-S_1 [19, 21] refined this approach by restricting to unit-norm vectors, introducing higher-order abstractions, and ensuring that terms of type qubit-to-qubit denote unitary maps. More recently, the Lambda-SX calculus [**?**] generalised Lambda-S to arbitrary non-entangled single-qubit bases, albeit through a purely syntactic framework restricted to first order. The calculus we present here extends these ideas to a unit-norm, higher-order setting over multiple-qubit bases, grounded in a realisability semantics.

The realisability methodology, originating with Kleene's work on Heyting arithmetic [28], provides a constructive framework that connects operational semantics with type systems. In our context, it allows the extraction of a sound type system directly from the reduction semantics of the calculus, ensuring that safety properties hold by construction. The approach proceeds as follows:

1. Define a calculus with a deterministic evaluation strategy.
2. Define types as sets of closed values in the language.
3. Define the typing judgement so that asserting that a term has a given type is, by definition, to state that the term reduces to a value of that type.

Each typing rule therefore corresponds to a provable theorem in this setting. Rather than building ad hoc typing rules, the system is derived from the computational content of the calculus, making it possible to define whole families of type systems by proving the validity of new rules.

Following this approach, we enrich abstractions with explicit basis decorations. Intuitively, the reduction system treats values expressed in the chosen basis as classical data, while linear combinations of these values represent quantum data and reduce linearly within the term. This refinement enables duplication and erasure for qubits in known bases while maintaining linear handling for unknown qubits.

The objective of this work is twofold. First, to employ the extracted type system to provide a more precise description of programs. Second, to take advantage of the extended syntax to express quantum algorithms in a flexible and compositional way, rather than merely translating circuits.

Related work. The idea of tracking basis information within quantum programming languages has appeared in several forms across the literature. The work in [30] proposed an abstract model for static entanglement analysis, where the basis of qubits is used to control duplication and track non-entanglement properties. The approach in [**?**] introduced a typed λ-calculus that incorporates basis annotations into the type system, ensuring linear handling of quantum data while providing strong meta-theoretical guarantees. Our system generalises these ideas by allowing abstractions to range over arbitrary—possibly entangled—bases, extending basis awareness beyond the single-qubit setting and recovering higher-order computation.

A number of other frameworks support reasoning about multiple bases through different paradigms. The green and red spiders in ZX-calculus correspond to computations along the canonical and diagonal bases respectively. The Many-Worlds Calculus [**?**] also accommodates superpositions of programs but within a diagrammatic semantics. By contrast, our approach operates directly within a typed term calculus, allowing basis transitions to be tracked syntactically.

From a categorical standpoint, the framework in [**?**] models measurement and decoherence through *quantum information effects*, capturing basis change as a semantic effect. Similarly, [**?**] introduces QuantumΠ, a language combining two interpretations of a reversible classical calculus—one per basis—via an effect construction. In both cases, basis sensitivity arises semantically. In contrast, λ_B integrates it directly at the syntactic and typing levels, ensuring that basis transformations are explicit in program structure.

Other languages aim to unify classical and quantum computation. For instance, [**?**] generalises classical control constructs and interprets duplication and discarding semantically as entanglement and partial trace. Our calculus follows the opposite philosophy: linearity and duplicability are syntactically restricted by the basis-sensitive type system, guaranteeing coherent quantum control without relying on external semantic constraints.

Finally, related efforts in semantic characterisation—such as the fully abstract model of [**?**] or the dual calculi described in [**?**]—approach the logical foundations of quantum computation from complementary angles. In contrast, λ_B focuses on the syntactic distinction of bases and superpositions within a unified quantum λ-calculus, providing a concrete basis-sensitive typing and higher-order quantum control.

Structure. The chapter is organised as follows. 3.2 introduces the core language, its syntax, congruence rules, and basis-dependent substitution. 3.3 defines the operational semantics, and 3.4 presents the realisability model with unitary type semantics, the characterisation of unitary operators, and the typing rules. 3.5 presents some representative examples. All omitted proofs appear in the appendices, and 3.6 concludes with final remarks and perspectives.

$$
\begin{array}{lll}
v ::= x \mid \lambda x_B \,.\, \vec{t} \mid (v, v) \mid |0\rangle \mid |1\rangle & & (\mathrm{V}) \\
t ::= v \mid tt \mid (t,t) \mid \mathsf{let}_{(B_1,B_2)}\ (x,y) = \vec{t}\ \mathsf{in}\ \vec{t} \mid \mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v} \mapsto \vec{t} \mid \cdots \mid \vec{v} \mapsto \vec{t}\} & & (\Lambda) \\
\vec{v} ::= v \mid \vec{v} + \vec{v} \mid \alpha \vec{v} \mid \vec{0} & (\alpha \in \mathbb{C}) & (\vec{\mathrm{V}}) \\
\vec{t} ::= t \mid \vec{t} + \vec{t} \mid \alpha \vec{t} \mid \vec{0} & (\alpha \in \mathbb{C}) & (\vec{\Lambda})
\end{array}
$$

Table 3.1: *Syntax of the calculus, where* $B, B_1, B_2 \subseteq \vec{\mathrm{V}}$.

3.2 Core language

3.2.1 Syntax and congruence

This section presents the calculus on which our realisability model will be built. It is a λ-calculus extended with linear combinations of terms, in the line of [**?**].

The syntax of the calculus is described in 3.1. The calculus is divided into four distinct syntactic categories: *basic values* (V), *basic terms* (Λ), *value distributions* ($\vec{\mathrm{V}}$), and *term distributions* ($\vec{\Lambda}$). Values are composed of variables, decorated lambda abstractions, and two basis values representing orthogonal vectors, $|0\rangle$ and $|1\rangle$. A pair of values is also considered a value. Terms include values, applications, pair constructors and destructors, and pattern matching for orthogonal vectors, represented by the case operator. Both term and value distributions are built as $\mathbb{C}$-linear combinations of terms and values, respectively. We use v, u, w to denote values and t, s, r for terms, writing $\vec{\cdot}$ when they are distributions.

The subsets B, B_1, and B_2 appearing in the abstractions and pair destructors denote the bases of the vector spaces in which the terms are expressed; their precise nature is made explicit below. Before formally defining the notion of bases, we first establish the algebraic structure underlying the space of value distributions. This is achieved by introducing the congruence relation defined in 3.2. The congruence captures the intended behaviour of addition and scalar multiplication, allowing us to treat linear combinations of terms and values as genuine algebraic entities rather than mere syntactic constructions. In particular, it enforces the commutativity and associativity of addition and the distributivity of scalar multiplication, thereby justifying summation notation $\sum^{\vec{}}$.

The rules in 3.2 ensure that the set of value distributions satisfies the axioms of a vector space. The notion of basis in this calculus builds on this algebraic foundation. Once the vector structure is established, we can define which subsets $B \subseteq \vec{\mathrm{V}}$ qualify as *bases*, thereby justifying the decorations appearing in the syntax of 3.1.

We will also consider the following notation for linear combinations of pairs. We stress that this notation for pairs does not appear in the syntax, but is rather useful to describe specific states.

$$(\alpha t + \vec{t_1}, \vec{t_2}) := \alpha(t, \vec{t_2}) + (\vec{t_1}, \vec{t_2}) \qquad (r, \alpha t + \vec{t_1}) := \alpha(r, t) + (r, \vec{t_1})$$

Before proceeding further, let us briefly illustrate the intuition behind the congruence in 3.2. The key idea is that arguments are decomposed over the bases associated with their corresponding abstractions. As in linear algebra, a vector can be rewritten as a linear combination of basis elements; for instance, $(1,0) = \frac{1}{\sqrt{2}}\left(\frac{(1,1)}{\sqrt{2}} + \frac{(1,-1)}{\sqrt{2}}\right)$ expresses $(1,0)$ in the basis $\{\frac{(1,1)}{\sqrt{2}}, \frac{(1,-1)}{\sqrt{2}}\}$. By identifying $(1,0)$ with $|0\rangle$ and $(0,1)$ with $|1\rangle$, we see that the calculus allows every vector (or value distribution) to be expressed as a superposition of elements of the basis attached to each abstraction. The congruence ensures that these linear

$$
\begin{array}{cc}
0\,\vec{t} \equiv \vec{0} & \vec{t} + \vec{0} \equiv \vec{t} \\
1\,\vec{t} \equiv \vec{t} & \alpha(\beta\,\vec{t}) \equiv (\alpha\beta)\,\vec{t} \\
\vec{t_1} + \vec{t_2} \equiv \vec{t_2} + \vec{t_1} & (\vec{t_1} + \vec{t_2}) + \vec{t_3} \equiv \vec{t_1} + (\vec{t_2} + \vec{t_3}) \\
(\alpha + \beta)\,\vec{t} \equiv \alpha\,\vec{t} + \beta\,\vec{t} & \alpha(\vec{t_1} + \vec{t_2}) \equiv \alpha\vec{t_1} + \alpha\vec{t_2} \\
\vec{t}\,(\alpha\,\vec{s}) \equiv \alpha(\vec{t}\,\vec{s}) & (\alpha\,\vec{t})\,\vec{s} \equiv \alpha(\vec{t}\,\vec{s}) \\
(\vec{t} + \vec{s})\,\vec{r} \equiv \vec{t}\,\vec{r} + \vec{s}\,\vec{r} & \vec{t}\,(\vec{s} + \vec{r}) \equiv \vec{t}\,\vec{s} + \vec{t}\,\vec{r}
\end{array}
$$

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

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

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

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

Table 3.2: *Term congruence*

combinations behave algebraically as in a complex vector space, supporting the standard operations of addition, scalar multiplication, and the zero vector.

To properly characterise the sets that decorate the lambda abstractions, we must first specify the kind of values they contain.

Definition 1 (Qubits)**.** *A* one-dimensional qubit *is a value distribution of the form* $\alpha\,|0\rangle + \beta\,|1\rangle$*, where* $|\alpha|^2 + |\beta|^2 = 1$*. An* n-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* α *and* β *satisfy the same normalisation condition.*

We use the usual Dirac shorthand $|xy\rangle$ for $(|x\rangle, |y\rangle)$, and extend it to longer tuples by associating to the right.

From now on, we call the value distributions, i.e. the elements of $\vec{\mathsf{V}}$, *vectors*. The vector space $\vec{\mathsf{V}}$ is equipped with an inner product defined by, $\langle \vec{v} \mid \vec{w} \rangle := \sum_{i=1}^{n} \sum_{j=1}^{m} \overline{\alpha_i}\,\beta_j\,\delta_{v_i,w_j}$, and an ℓ_2-norm $\|\vec{v}\| := \sqrt{\langle \vec{v} \mid \vec{v} \rangle} = \sqrt{\sum_{i=1}^{n} |\alpha_i|^2}$, where $\vec{v} = \sum_{i=1}^{n} \alpha_i v_i$ and $\vec{w} = \sum_{j=1}^{m} \beta_j w_j$, and δ_{v_i,w_j} is the Kronecker delta, equal to 1 if $v_i = w_j$ and 0 otherwise.

With this notion of inner product, we can complete the description of the calculus syntax. As expected, two values are *orthogonal* when their inner product is equal to zero. We can now formally define the sets that decorate abstractions.

Definition 2 (Basis)**.** *A set of value distributions* B *is an* n-dimensional orthonormal basis *if it satisfies:*

1. *Each element of* B *is an* n*-dimensional qubit* (cf. 1)*.*
2. *Distinct elements of* B *are pairwise orthogonal.*

From now on, the syntax introduced in 3.1 is restricted to sets B forming n-dimensional orthonormal bases.

Unlike standard orthonormal bases, we require that their elements be qubits (rather than variables or abstractions). These sets indicate the basis in which a term is expressed:

qubits in the decorating basis are treated call-by-value, while others are decomposed as $\mathbb{C}$-linear combinations of basis elements, with the function applied linearly to each component. If a term cannot be expressed in the decorating basis, evaluation becomes stuck.

As in classical linear algebra, no non-trivial linear combination of basis elements yields the null vector; otherwise, some basis vector would violate pairwise orthogonality. Consequently, each decomposition over a basis is unique.

Theorem 1 (Unique decomposition)**.** *If B is an n-dimensional basis, then every n-dimensional qubit has a unique decomposition over B.* □

Corollary 1 (Preservation under congruence)**.** *If $\vec{v} \equiv \vec{w}$, then they share the same decomposition over any basis B.* □

We denote the computational basis by $\mathbb{B} = \{|0\rangle, |1\rangle\}$ and the diagonal basis by $\mathbb{X} = \{|+\rangle, |-\rangle\}$, where $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$.

3.2.2 Basis-dependent substitutions

As usual, the expression $\vec{t}\,[\vec{v}/x]$ denotes the usual capture-avoiding substitution of $\vec{v}$ for x in $\vec{t}$. However, beta-reduction depends on the basis chosen for the abstraction, so we must define a substitution that takes this mechanism into account. Intuitively, this operation substitutes variables with vectors expressed in the chosen basis; the accompanying coefficients are those of the value distribution being substituted.

Alongside this substitution, we expand the set of orthogonal basis with a special symbol, denoted $\mathcal{P}$. This symbol changes the modality of the lambda abstraction, allowing it to take other abstractions as arguments. This is the only instance of higher-order functions in the calculus. The $\mathcal{P}$, is a purely syntactical device, intuitively it can be thought as the canonical basis.

Definition 3 (Basis-dependent substitution)**.** *Let $\vec{t}$ be a term distribution, $\vec{v}$ a value distribution, x a variable, and B an orthonormal basis. We define the substitution $\vec{t}\,\langle\vec{v}/x\rangle_B$ as follows:*

$$\vec{t}\,\langle\vec{v}/x\rangle_B \;=\; \begin{cases} \displaystyle\sum_{i\in I} \alpha_i\ \vec{t}\,[\vec{b_i}/x] & \text{if } B = \{\vec{b_i}\}_{i\in I} \text{ and } \vec{v} \equiv \displaystyle\sum_{i\in I} \alpha_i\, \vec{b_i}, \\ \displaystyle\sum_{i\in I} \alpha_i\ \vec{t}\,[v_i/x] & \text{if } B = \mathcal{P} \text{ and } \vec{v} = \displaystyle\sum_{i\in I} \alpha_i\, v_i, \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

The two cases in the definition capture distinct substitution modes. In the first, substitution proceeds linearly using the decomposition of $\vec{v}$ over the explicit basis B. In the second, when $B = \mathcal{P}$, substitution proceeds linearly over the basic values that form $\vec{v}$. This case recovers the substitution mechanism of non base-sensitive calculi—as first defined in [**?**]—but generalises it by introducing basis-dependent behaviour. This special case is also the only one applicable to λ-abstractions, since abstractions do not belong to orthonormal bases.

The definition also extends to pairs of values. If $\vec{v} = \sum_{i\in I} \alpha_i\, (\vec{v_i}, \vec{w_i})$, with $\vec{v_i} \in \mathsf{Span}(B_1)$ and $\vec{w_i} \in \mathsf{Span}(B_2)$, then

$$\vec{t}\,\langle\vec{v}/x \otimes y\rangle_{B_1\otimes B_2} \;=\; \sum_{i\in I} \alpha_i \left(\vec{t}\,\langle\vec{v_i}/x\rangle_{B_1}\langle\vec{w_i}/y\rangle_{B_2}\right),$$

where B_1 and B_2 are (orthonormal) bases—or $\mathcal{P}$—associated with x and y, respectively; the symbol $\otimes$ in $B_1 \otimes B_2$ is purely notational.

Example 3.1. *Let* $\vec{v} = \alpha\,|01\rangle + \beta\,|10\rangle$. *Then*

$$(y,x)\langle \vec{v}/x \otimes y\rangle_{\mathbb{B}\otimes\mathbb{B}} = \alpha\,(y,x)[|0\rangle\,/x][|1\rangle\,/y] + \beta\,(y,x)[|1\rangle\,/x][|0\rangle\,/y] = \alpha\,|10\rangle + \beta\,|01\rangle$$

With this substitution in place, we can establish certain needed properties. First, basis-dependent substitution distributes over linear combinations.

Lemma 8 (Distributivity over linear combinations)**.** *For term distributions* $\vec{t_i}$*, a value distribution* $\vec{v}$*, a variable* x*, coefficients* $\alpha_i \in \mathbb{C}$*, and a basis* B *such that* $\langle \vec{v}/x\rangle_B$ *is defined:* $\left(\sum_i \alpha_i \vec{t_i}\right)\langle \vec{v}/x\rangle_B \equiv \sum_i \alpha_i \vec{t_i}\langle \vec{v}/x\rangle_B$. □

The next result states that substitution behaves consistently within each equivalence class induced by the congruence $\equiv$.

Lemma 9 (Compatibility with congruence)**.** *For value distributions* $\vec{v}, \vec{w}$*, a term distribution* $\vec{t}$*, and an orthonormal basis* B *such that both* $\langle \vec{v}/x\rangle_B$ *and* $\langle \vec{w}/x\rangle_B$ *are defined: if* $\vec{v} \equiv \vec{w}$*, then* $\vec{t}\langle \vec{v}/x\rangle_B = \vec{t}\langle \vec{w}/x\rangle_B$. □

We remark that the result in the previous lemma uses $=$ instead of $\equiv$ since $\vec{t}\langle \vec{v}/x\rangle_B$ and $\vec{t}\langle \vec{w}/x\rangle_B$ are both substitution with the exact same outcome. Note that this property does not extend across different bases; that is, $\vec{t}\langle \vec{v}/x\rangle_A \not\equiv \vec{t}\langle \vec{v}/x\rangle_B$. For example,

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

This difference arises because the relation $\equiv$ does not commute with lambda abstraction, nor with the case construct. Although the two terms are operationally equivalent, the calculus distinguishes between the superposition of results, $\lambda x_B\,.\,\alpha\vec{v_1} + \beta\vec{v_2}$, and the superposition of functions, $\alpha(\lambda x_B\,.\,\vec{v_1}) + \beta(\lambda x_B\,.\,\vec{v_2})$. This distinction reflects a physical intuition: the former corresponds to a single experiment producing a superposition of outcomes, while the latter represents a superposition of distinct experiments.

Finally, we introduce a convenient notation for generalised substitutions over a term by closed values. A substitution σ can be seen as a finite set of individual substitutions applied consecutively to a term. Formally, for a term $\vec{t}$, closed value distributions $\vec{v_1}, \ldots, \vec{v_n}$, variables $x_1, \ldots, x_n$, and bases $B_1, \ldots, B_n$:

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

Since each $\vec{v_i}$ is closed, the order of substitutions is irrelevant. We regard σ as a partial function from variables to pairs of closed value distributions and bases, and write $\mathrm{dom}(\sigma)$ for its domain. The operation extends naturally: for a term $\vec{t}$, substitution σ, a new variable $x \notin \mathrm{dom}(\sigma)$, value distribution $\vec{v}$, and basis B, $\vec{t}\langle\sigma\rangle\langle \vec{v}/x\rangle_B = \vec{t}\langle\sigma'\rangle$, where σ' extends σ by mapping x to $(\vec{v}, B)$. Likewise, two disjoint substitutions σ_1 and σ_2 can be merged: $\vec{t}\langle\sigma_1\rangle\langle\sigma_2\rangle = \vec{t}\langle\sigma'\rangle$, where $\mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2) = \emptyset$ and σ' coincides with σ_i on $\mathrm{dom}(\sigma_i)$ for $i = 1, 2$.

3.3 Operational semantics

The reduction system interprets every vector relative to the basis attached to its abstraction, allowing a step only when the argument can be decomposed on that basis. 3.3 defines the elementary relation $\rightsquigarrow$, while terms are considered modulo the congruence of 3.2. Hence the effective reduction, written $\longrightarrow$, is defined modulo $\equiv$: a step $\vec{t} \longrightarrow \vec{r}$ abbreviates $\vec{t} \equiv \vec{t}' \rightsquigarrow \vec{r}' \equiv \vec{r}$.

The side condition $\alpha \neq 0$ avoids vacuous steps such as $0t \rightsquigarrow 0r$, preventing spurious nondeterminism. On that note, the condition on the contextual rule of the sum, imposes a strategy preserving the determinism of $\rightsquigarrow$.

$$\text{If } \vec{t}\langle\vec{v}/x\rangle_X \text{ is defined,} \quad (\lambda x_X \,.\, \vec{t})\,\vec{v} \rightsquigarrow \vec{t}\langle\vec{v}/x\rangle_X$$

$$\text{If } \vec{t}\langle\vec{v}/x\rangle_{X\otimes Y} \text{ is defined,} \quad \mathsf{let}_{(X,Y)}\ (x,y) = \vec{v}\ \mathsf{in}\ \vec{t} \rightsquigarrow \vec{t}\langle\vec{v}/x\otimes y\rangle_{X\otimes Y}$$

$$\mathsf{case}\ \sum_{i=1}^{n} \alpha_i \vec{v_i}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{t_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{t_n}\} \rightsquigarrow \sum_{i=1}^{n} \alpha_i \vec{t_i}$$

$$\frac{t \rightsquigarrow \vec{r}}{st \rightsquigarrow s\vec{r}} \qquad \frac{t \rightsquigarrow \vec{r}}{tv \rightsquigarrow \vec{r}\,v} \qquad \frac{t \rightsquigarrow \vec{r} \quad \alpha \neq 0}{\alpha t \rightsquigarrow \alpha \vec{r}}$$

$$\frac{t_j \rightsquigarrow \vec{r}_j \quad \text{With } \vec{v}_i \text{ values and } t_j < t_i \text{ in the lexicographical order}}{(\sum_{i=1}^{j-1} \vec{v}_i) + \alpha_j t_j + (\sum_{i=j+1}^{n} \alpha_i \vec{t}_i) \rightsquigarrow (\sum_{i=1}^{j-1} \vec{v}_i) + \alpha_j r_j + (\sum_{i=j+1}^{n} \alpha_i \vec{t}_i)}$$

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

$$\frac{t \rightsquigarrow \vec{r}}{\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v} \mapsto \vec{s_1} \mid \cdots \mid \vec{w} \mapsto \vec{s_n}\} \rightsquigarrow \mathsf{case}\ \vec{r}\ \mathsf{of}\{\vec{v} \mapsto \vec{s_1} \mid \cdots \mid \vec{w} \mapsto \vec{s_n}\}}$$

Table 3.3: *Reduction system*

The main reduction rules are β-reduction, let binding, and case pattern matching. Both λ and let bind variables decorated with an orthonormal basis, indicating which vectors are treated as classical data. Linear combinations of these vectors are handled as quantum data, reducing linearly by the congruence rules of 3.2, so that $t(\alpha\vec{s} + \beta\vec{r})$ is equivalent to $\alpha\, t\vec{s} + \beta\, t\vec{r}$.

The only exception occurs for higher-order terms: since no orthogonal bases are defined for abstractions, we introduce a special modality $\mathcal{P}$ reducing linearly over the basis values of a distribution. For instance,

$$(\lambda x_{\mathcal{P}} \,.\, \vec{t}\,) \sum_{i=1}^{n} \alpha_i (\lambda y_X \,.\, \vec{s_i}) \longrightarrow \sum_{i=1}^{n} \alpha_i\, \vec{t}\,[\lambda y_X \,.\, \vec{s_i}/x].$$

The case pattern matching controls program flow. Each operator keeps track of a set of orthogonal values and tests whether the argument equals each vector, selecting the matching branch. If the argument is a linear combination of several vectors, the result is the corresponding linear combination of the branches. For example:

$$\mathsf{case}\ |-\rangle\ \mathsf{of}\{|0\rangle \mapsto \vec{t_1} \mid |1\rangle \mapsto \vec{t_2}\} \longrightarrow \tfrac{1}{\sqrt{2}}\,\vec{t_1} - \tfrac{1}{\sqrt{2}}\,\vec{t_2}.$$

The advantage over a conditional branching is the ability to match against several vectors simultaneously. For boolean tuples this makes no difference, as each component can be treated independently. However, some orthogonal bases cannot be expressed as products of smaller ones. This general case allows us to match directly against those vectors. For example, using the *Bell basis*[1]:

$$\mathsf{case}\ \vec{v}\ \mathsf{of}\ \{\Phi^+ \mapsto \vec{t_1}\ \mid \Phi^- \mapsto \vec{t_2}\ \mid \Psi^+ \mapsto \vec{t_3}\ \mid \Psi^- \mapsto \vec{t_4}\ \}.$$

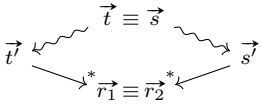
[1]The four Bell states are $\Phi^{\pm} = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle)$ and $\Psi^{\pm} = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)$.

This Bell basis is central in quantum communication. In **??**, we revisit the quantum teleportation protocol, which relies heavily on these states.

Defining the system in this way yields a strategy within the *call-by-value* family, namely a generalisation of the *call-by-basis* strategy introduced in [**?**] and further analysed in [**?**]. Whereas call-by-basis fixes a single computational basis for evaluation, our variant, which we call *call-by-arbitrary-basis*, allows each abstraction to attach its own orthonormal basis to its argument. Evaluation remains weak: no reduction takes place under λ, pairs, `let`, or `case` constructors.

The congruence relation on terms gives rise to different redexes. We write $\twoheadrightarrow$ for the reflexive–transitive closure of $\longrightarrow$. We can show that the equivalence relation $\equiv$ commutes with $\twoheadrightarrow$; in other words, equivalence is preserved by reduction modulo $\equiv$.

Theorem 2 (Reduction preserves equivalence)**.** *Let $\vec{t}$ and $\vec{s}$ be closed term distributions with $\vec{t} \equiv \vec{s}$. If $\vec{t} \rightsquigarrow \vec{t'}$ and $\vec{s} \rightsquigarrow \vec{s'}$, 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:*



□

□

Remark 1. *Since the reduction relation $\twoheadrightarrow$ is defined modulo $\equiv$, the result above is equivalent to stating that there exists a single distribution $\vec{r}$ such that $\vec{t'} \twoheadrightarrow \vec{r}$ and $\vec{s'} \twoheadrightarrow \vec{r}$. Moreover, as the elementary reduction $\rightsquigarrow$ is deterministic, the reduction relation $\longrightarrow$ is also deterministic; that is, if $\vec{t} \longrightarrow \vec{r_1}$ and $\vec{t} \longrightarrow \vec{r_2}$, then $\vec{r_1} \equiv \vec{r_2}$.*

3.4 Realizability model

3.4.1 Unitary type semantics

Given the deterministic machine presented in the previous section (see 1), the next step towards extracting a typing system is to define the sets of values that characterise its types. To achieve this, we first need to identify the notion of a type.

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

Definition 4 (Unitary value distribution)**.** *A value distribution $\vec{v}$ is said to be* unitary *if its norm equals 1, that is, if $\vec{v} \in \mathcal{S}_1$.*

Definition 5 (Unitary type)**.** *A* unitary type *(or simply a* type*) is a notation A together with a set of unitary value distributions, denoted $[\![A]\!]$, called the* unitary semantics *of A.*

We now turn to type realizers. Since the global phase of a quantum state has no physical significance, terms that differ only by a phase should share the same type. Thus, we assign identical types to $\vec{t}$ and $e^{i\theta}\vec{t}$, a principle that guides the definition of realizers.

Definition 6 (Type realizer)**.** *Given a type A and a term distribution $\vec{t}$, we say that $\vec{t}$* realizes *A (written $\vec{t} \Vdash A$) if there exists a value distribution $\vec{v} \in [\![A]\!]$ such that $\vec{t} \twoheadrightarrow e^{i\theta}\vec{v}$ for some $\theta \in \mathtt{R}$,*

$$A := B_X \mid A \Rightarrow A \mid A \times A \mid \sharp A, \qquad \text{where } X \text{ is any orthonormal basis.}$$

$$\begin{aligned}
[\![B_X]\!] &:= X \\
[\![A \times B]\!] &:= \{\, (\vec{v}, \vec{w}) \;\mid\; \vec{v} \in [\![A]\!],\ \vec{w} \in [\![B]\!] \,\} \\
[\![A \Rightarrow B]\!] &:= \{\, \sum_{i=1}^{n} \alpha_i (\lambda x_X \,.\, \vec{t_i}) \in \mathcal{S}_1 \;\mid\; \forall \vec{w} \in [\![A]\!],\ (\sum_{i=1}^{n} \alpha_i \vec{t_i}) \langle \vec{w}/x \rangle_X \Vdash B \,\} \\
[\![\sharp A]\!] &:= ([\![A]\!]^{\perp})^{\perp} \quad \text{Where: } A^{\perp} = \{ \vec{v} \in \mathcal{S}_1 \mid \langle \vec{v} \mid a \rangle = 0, \forall a \in A \}
\end{aligned}$$

Table 3.4: *Type notations and semantics*

With the notions of unitary types and their realizers in place, we can now define a concrete approach for our language. We begin with the type grammar given in 3.4 and build a simple algebra from the sets of values we aim to represent. Before that, we introduce the notion of orthogonal complement, which will be used in the semantics of the $\sharp$ type:

$$A^{\perp} = \{\, \vec{v} \in \mathcal{S}_1 \mid \langle \vec{v} \mid a \rangle = 0 \text{ for all } a \in A \,\}.$$

The types B_X serve as atomic types. Each of them represents a finite set X of orthogonal vectors forming an orthonormal basis. For instance, a boolean type can be represented by a basis of size 2, yet we are not restricted to a single one, since there are infinitely many bases to choose from.

The type $A \times B$ corresponds to the cartesian product of A and B. However, since the syntax admits pairs only of basic values, the semantic clause in 3.4 should be read as an abbreviation for the following expanded form:

$$[\![A \times B]\!] := \Big\{ \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j (v_i, w_j) \;\mid\; \sum_{i=1}^{n} \alpha_i v_i \in [\![A]\!],\ \sum_{j=1}^{m} \beta_j w_j \in [\![B]\!] \Big\}.$$

We stress this for rigour, but for readability we shall continue using the concise notation of pairs. The arrow type $A \Rightarrow B$ consists of distributions of λ-abstractions that map elements of $[\![A]\!]$ to realizers of B. Finally, the type $\sharp A$ denotes the double orthogonal complement of A intersected with the unit sphere. It represents quantum data—linear resources that cannot be erased or duplicated. Intuitively, applying the $\sharp$ operator to a type A yields the span of the original interpretation (intersected with the unit sphere). This captures the possible linear combinations of values in the unitary semantics of A, as stated in the following theorem.

Theorem 3. *The interpretation of a type $\sharp A$ contains precisely the norm-1 linear combinations of values in $[\![A]\!]$:* $[\![\sharp A]\!] = ([\![A]\!]^{\perp})^{\perp} = \mathsf{Span}([\![A]\!]) \cap \mathcal{S}_1$. □

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:

$$\langle \vec{v} \mid \vec{w} \rangle = \langle \sum_{i=1}^{n} \alpha_i \vec{v_i} \mid \vec{w} \rangle$$

$$= \sum_{i=1}^{n} \overline{\alpha_i} \langle \vec{v_i} \mid \vec{w} \rangle = 0$$

Then $\vec{v} \in (\llbracket A \rrbracket^\perp)^\perp$.

$(\llbracket A \rrbracket^\perp)^\perp \subseteq \mathsf{Span}(\llbracket A \rrbracket) \cap \mathcal{S}_1$**:** Reasoning by contradiction, we assume that there is a $\vec{v} \in (\llbracket A \rrbracket^\perp)^\perp$ such that $v \notin \mathsf{Span}(\llbracket A \rrbracket) \cap \mathcal{S}_1$. Since $\vec{v} \notin \mathsf{Span}(\llbracket A \rrbracket)$, $\vec{v} = \vec{w_1} + \vec{w_2}$ such that $\vec{w_1} \in \mathsf{Span}\llbracket A \rrbracket$ and $\vec{w_2}$ is a non-null vector which cannot be written as a linear combination of elements of $\llbracket A \rrbracket$. In other words, $\vec{w_2} \in \llbracket A \rrbracket^\perp$. 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 $\vec{v} \notin (\llbracket A \rrbracket^\perp)^\perp$. The contradiction stems from assuming $\vec{v} \notin \mathsf{Span}\llbracket A \rrbracket \cap \mathcal{S}_1$. □

The following theorem shows that, as expected for a span, multiple applications of the $\sharp$ operator have no further effect beyond the first application.

Theorem 4. *The $\sharp$ operator is idempotent; that is, $\llbracket \sharp A \rrbracket = \llbracket \sharp(\sharp A) \rrbracket$.* □

Proof. We want to prove that $((((\llbracket A \rrbracket^\perp)^\perp)^\perp)^\perp = (\llbracket A \rrbracket^\perp)^\perp$. 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: $\llbracket \sharp(\sharp A) \rrbracket = A^{\perp^4} = A^{\perp^2} = \llbracket \sharp A \rrbracket$ □

Remark 2. *A basis type B_X may consist of value distributions of pairs and can therefore be written as the product type of smaller bases. For example, if $X = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, then $\llbracket B_X \rrbracket = \llbracket B_\mathbb{B} \times B_\mathbb{B} \rrbracket$. However, this is not possible for entangled bases. A clear example is the Bell basis.*

It remains to verify that our type algebra indeed captures the intended sets of value distributions. The following theorem shows that every member of a type interpretation has norm 1.

Theorem 5. *For every type A, $\llbracket A \rrbracket \subseteq \mathcal{S}_1$.* □

Defining types as sets of values naturally induces a semantic notion of subtyping. We say that a type A is a subtype of a type B (written $A \leq B$) when the set of realizers of A is included in that of B. If the two sets coincide, we say that A and B are *isomorphic* (written $A \cong B$).

Example 3.2. *For every type A, we have $A \leq \sharp A$. For the base types $B_\mathbb{B}$ and $B_\mathbb{X}$, however, neither inclusion holds: $B_\mathbb{B} \not\leq B_\mathbb{X}$ and $B_\mathbb{X} \not\leq B_\mathbb{B}$. Nevertheless, their linear extensions coincide, since $\sharp B_\mathbb{B} \cong \sharp B_\mathbb{X}$.*

Although every type is defined by norm-1 value distributions, not every norm-1 distribution is contained in the interpretation of a type. For example, consider the distribution $\frac{1}{\sqrt{2}}(|0\rangle + |00\rangle)$. Another example is a linear combination of abstractions defined over different bases. For instance, the term

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

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 in turn belongs to the set $\llbracket B_\mathbb{B} \Rightarrow B_\mathbb{X} \rrbracket$.

We denote by $\mathbb{T}$ the set of all types and by $\mathbb{T}_B$ the set of all basis types B_X.

3.4.2 Characterization of unitary operators

One of the main results of [19] is the characterisation of $\mathbb{C}^2 \to \mathbb{C}^2$ unitary operators using values in $[\![\sharp B_{\mathbb{B}} \Rightarrow \sharp B_{\mathbb{B}}]\!]$ [19, Theorem IV.12]. In this subsection we extend this result. Our goal is to prove that abstractions of type $\sharp B_X \Rightarrow \sharp B_Y$, where both bases have size n, represent unitary operators $\mathbb{C}^n \to \mathbb{C}^n$.

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

Lemma 10. *Let X and Y be orthonormal bases of the same finite dimension, and let $\lambda x_X \,.\, \vec{t}$ be a closed λ-abstraction. Then $\lambda x_X \,.\, \vec{t} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$ if and only if for all $\vec{v_i}, \vec{v_j} \in [\![B_X]\!]$, there exist value distributions $\vec{w_i}, \vec{w_j} \in [\![\sharp B_Y]\!]$ such that, $\vec{t}[\vec{v_i}/x] \twoheadrightarrow \vec{w_i}$ and $\vec{t}[\vec{v_j}/x] \twoheadrightarrow \vec{w_j}$, with $\vec{w_i} \perp \vec{w_j}$ whenever $i \neq j$.* □

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:*

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

Terms such as $|0\rangle$ and $|1\rangle$ are syntactic objects of the calculus, not vectors of $\mathbb{C}^2$. Nevertheless, when discussing the behaviour of terms on Hilbert spaces, we shall occasionally abuse notation and identify value distributions representing qubits with their corresponding canonical basis vectors in $\mathbb{C}^n$. This identification applies only to those distributions that denote quantum data, not to general syntactic values such as λ-abstractions—in particular, to all elements of $[\![\sharp B_X]\!]$ for any orthonormal basis X. This identification can be made precise as follows.

Definition 9. *Let X be an orthonormal basis of size n. For every $\vec{v} \in X$, we can write $\vec{v} \equiv \sum_{i=1}^{n} \alpha_i \, |i\rangle$, where $|i\rangle$ denotes the n-qubit tuple of $|0\rangle$ and $|1\rangle$ corresponding to the binary representation of i, and $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. For example, for $n = 4$, $|3\rangle$ is $|0011\rangle$. We then define $\pi_n : X \to \mathbb{C}^n$ by $\pi_n(\vec{v}) = (\alpha_1, \ldots, \alpha_n)$.*

To lighten notation, we shall henceforth omit π_n and use $\vec{v}$ directly.

We now establish a correspondence between λ-abstractions and operators on $\mathbb{C}^n$. Intuitively, an abstraction represents a linear operator when its operational behaviour coincides with the action of that operator on vectors. Formally:

Definition 10. *A λ-abstraction $\lambda x_X \,.\, \vec{t}$ is said to* represent *an operator $F : \mathbb{C}^n \to \mathbb{C}^n$ if $(\lambda x_X \,.\, \vec{t})\,\vec{v} \twoheadrightarrow \vec{w}$ if and only if $F(\vec{v}) = \vec{w}$.*

This definition, together with 10, allows us to build a characterisation of unitary operators as values in $[\![\sharp B_X \Rightarrow \sharp B_Y]\!]$.

Theorem 6 (Characterisation of Unitary Operators)**.** *Let X and Y be orthonormal bases of size n. A closed λ-abstraction $\lambda x_X \,.\, \overrightarrow{t} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$ if and only if it represents a unitary operator $F : \mathbb{C}^n \to \mathbb{C}^n$.*

Proof. Necessity. Suppose that $\lambda x_X \,.\, \overrightarrow{t} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$. Then, by 10, for every $\overrightarrow{v_i} \in [\![B_X]\!]$ there exists $\overrightarrow{w_i} \in [\![\sharp B_Y]\!]$ such that $\overrightarrow{t}[\overrightarrow{v_i}/x] \twoheadrightarrow \overrightarrow{w_i}$ and $\overrightarrow{w_i} \perp \overrightarrow{w_j}$ whenever $i \neq j$. Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be the operator defined by $F(\overrightarrow{v_i}) = \overrightarrow{w_i}$. By linearity over X, it follows that $\lambda x_X \,.\, \overrightarrow{t}$ represents F. Moreover, F is unitary since $\|\overrightarrow{w_i}\|_{\mathbb{C}^n} = \|\overrightarrow{w_j}\|_{\mathbb{C}^n} = 1$ and $\langle \overrightarrow{w_i} \mid \overrightarrow{w_j} \rangle_{\mathbb{C}^n} = 0$.

Sufficiency. Conversely, suppose that $\lambda x_X \,.\, \overrightarrow{t}$ represents a unitary operator $F : \mathbb{C}^n \to \mathbb{C}^n$. Then, for each $\overrightarrow{v_i} \in [\![B_X]\!]$, there exists $\overrightarrow{w_i} \in [\![B_Y]\!]$ such that $F(\overrightarrow{v_i}) = \overrightarrow{w_i}$ and $(\lambda x_X \,.\, \overrightarrow{t})\overrightarrow{v_i} \twoheadrightarrow \overrightarrow{w_i}$. Hence, $(\lambda x_X \,.\, \overrightarrow{t})\overrightarrow{v_i} \rightsquigarrow \overrightarrow{t}\langle \overrightarrow{v_i}/x \rangle_X = \overrightarrow{t}[\overrightarrow{v_i}/x] \twoheadrightarrow \overrightarrow{w_i} \in [\![\sharp B_Y]\!]$, since $\|\overrightarrow{w_i}\| = \|F(\overrightarrow{v_i})\|_{\mathbb{C}^n} = 1$. From 10, it follows that $\lambda x_X \,.\, \overrightarrow{t} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$, and moreover $\langle \overrightarrow{w_i} \mid \overrightarrow{w_j} \rangle = \langle F(\overrightarrow{v_i}) \mid F(\overrightarrow{v_j}) \rangle_{\mathbb{C}^n} = 0$. □

This result naturally extends to unitary distributions of λ-abstractions, since a term of the form $\lambda x_X \,.\, \sum_{i=1}^n \alpha_i \overrightarrow{t_i}$ is syntactically different but computationally equivalent to $\sum_{i=1}^n \alpha_i \lambda x_X \,.\, \overrightarrow{t_i}$. Hence, the characterisation of unitary operators also applies to superpositions of abstractions sharing the same basis X.

3.4.3 Typing rules

In this section, we focus on enumerating and proving the validity of various typing rules. The objective is to extract a reasonable set of rules that can constitute a type system. We first need to lay the groundwork to properly define what it means for a typing rule to be valid.

Definition 11. *A* context *(denoted by capital Greek letters Γ, Δ) is a finite mapping $\Gamma : \mathsf{Var} \to \mathbb{T} \times \mathbb{T}_B$ assigning a type and a basis to each variable in its domain. We write $\Gamma = x_1{}^{X_1} : A_1, \cdots, x_n{}^{X_n} : A_n$ to indicate that $\Gamma(x_i) = (A_i, B_{X_i})$ for each i.*

As in standard typing judgements, the context records the types of a term's free variables. However, since substitution in our calculus depends on a basis, we also wish to record that information. This is not strictly necessary, as the basis with respect to which a variable is interpreted should not affect its type. For instance, consider the following substitutions:

$$(\lambda x_{\mathbb{B}} \,.\, (x, y))\langle |0\rangle \,/y\rangle_{\mathbb{B}} = \lambda x_{\mathbb{B}} \,.\, (x, |0\rangle),$$
$$\text{and} \quad (\lambda x_{\mathbb{B}} \,.\, (x, y))\langle |0\rangle \,/y\rangle_{\mathbb{X}} = \tfrac{1}{\sqrt{2}}\big((\lambda x_{\mathbb{B}} \,.\, (x, |+\rangle)) + (\lambda x_{\mathbb{B}} \,.\, (x, |-\rangle))\big).$$

These terms are not syntactically identical, yet they are computationally equivalent. Since typing via realisability captures computational behaviour, their types coincide. Nevertheless, we retain basis information in contexts, as it will simplify the forthcoming proofs.

Definition 12. *Given a context Γ, its* unitary semantics*, denoted $[\![\Gamma]\!]$, is the set of substitutions defined by*

$$[\![\Gamma]\!] := \{\sigma \textit{ substitution} \mid \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma) \textit{ and } \forall x_i \in \mathrm{dom}(\Gamma),$$
$$\Gamma(x_i) = (A_i, B_{X_i}) \textit{ implies } \sigma(x_i) = \langle \overrightarrow{v_i}/x_i \rangle_{X_i} \textit{ for some } \overrightarrow{v_i} \in [\![A_i]\!]\}.$$

To ensure a coherent treatment of quantum data, we must guarantee that qubits are handled linearly. The first step is to identify which variables in the context represent quantum data—those associated with a type of the form $\sharp A$. We call the subset of Γ composed of such variables its *strict domain*.

Definition 13. *The* strict domain *of a context* Γ*, denoted* $\mathrm{dom}^\sharp(\Gamma)$*, is defined as* $\mathrm{dom}^\sharp(\Gamma) := \{x \in \mathrm{dom}(\Gamma) \mid [\![\Gamma(x)]\!] = [\![\sharp(\Gamma(x))]\!]\}$.

This definition relies on the idempotence of the $\sharp$ operator (4).

A typing judgement $\Gamma \vdash \vec{t} : A$ is valid if it satisfies two conditions. First, every free variable of $\vec{t}$ must belong to the domain of Γ, and every variable in the strict domain $\mathrm{dom}^\sharp(\Gamma)$ must occur in $\vec{t}$. This ensures that no information is erased and that all variables are properly accounted for. The linear treatment of quantum data is thus enforced by substitution.

Second, for every substitution in the unitary semantics of Γ, applying it to $\vec{t}$ must yield a term that reduces to a realizer of type A. This condition ensures that the operational behaviour of the term within the context is faithfully captured by the type. Formally:

Definition 14 (Typing judgement)**.** *A* typing judgement $\Gamma \vdash \vec{t} : A$ *is valid when:* $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$ *and for all* $\sigma \in [\![\Gamma]\!]$, $\vec{t}\langle\sigma\rangle \Vdash A$.

We are also interested in *orthogonal terms*, that is, terms that reduce to orthogonal values. We therefore introduce the following notion.

Definition 15. *An* orthogonality judgement $\Gamma \vdash (\Delta_1 \vdash \vec{t}) \perp (\Delta_2 \vdash \vec{s}) : A$ *is said to be* valid *when*

- *the judgements* $\Gamma, \Delta_1 \vdash \vec{t} : A$ *and* $\Gamma, \Delta_2 \vdash \vec{s} : A$ *are valid, and*
- *for for every* $\sigma \in [\![\Gamma, \Delta_1]\!]$ *and* $\tau \in [\![\Gamma, \Delta_2]\!]$*, there exist 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* $\vec{v} \perp \vec{w}$.

When both Δ_1 *and* Δ_2 *are empty, we just write* $\Gamma \vdash \vec{t} \perp \vec{s} : A$.

With these definitions in mind, a typing rule is *valid* when valid premises entail a valid conclusion. Although there are infinitely many valid rules (each corresponding to a theorem), 3.5 presents a representative subset forming a reasonable core typing system for the calculus, whose validity is stated below.

Theorem 7. *All the typing rules in 3.5 are valid.* □

The usual safety properties follow straightforwardly in this framework. *Confluence* is an immediate consequence of the reduction being deterministic (cf. 1). *Strong normalisation* follows directly from the definition of a realizer (cf. 6). *Subject reduction* is also immediate: indeed, if $\Gamma \vdash \vec{t} : A$ and $\vec{t} \to \vec{u}$, then $\Gamma \vdash \vec{u} : A$ by definition. However, if we restrict ourselves to a subset of typing rules—such as those presented in 3.5—we must ensure that this restricted system still suffices to type all reducts of a term, once the underlying realisability semantics is abstracted away. In our case, the rules proven valid in 7 suffice to guarantee subject reduction, as stated below.

Theorem 8 (Subject reduction)**.** *If* $\Gamma \vdash \vec{t} : A$ *can be derived using the set of rules in 3.5 and* $\vec{t} \to \vec{u}$*, then* $\Gamma \vdash \vec{u} : A$ *can also be derived by the same set of rules.* □

3.5 Examples

3.5.1 Deutsch's algorithm

We begin with *Deutsch's algorithm*, a canonical example that highlights how basis types in the λ_B calculus yield more informative typings for quantum programs. The algorithm is as follows. We are given black-box access to an *oracle* U_f that implements an unknown Boolean function $f : \{0,1\} \to \{0,1\}$. The oracle can only be either *constant* (both inputs map to the same output) or *balanced* (the two outputs differ). Classically, determining

$$\frac{B_X \leq A \text{ or } X = \mathcal{P}}{x^X : A \vdash x : A}\ (\text{Axiom}) \qquad \frac{\Gamma, x^X : A \vdash \sum_{i=1}^n \alpha_i \vec{t_i} : B}{\Gamma \vdash \sum_{i=1}^n \alpha_i (\lambda x_X \,.\, \vec{t_i}) : A \Rightarrow B}\ (\text{UnitLam})$$

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

$$\frac{\Gamma \vdash \vec{t} : A \times B \quad \Delta, x^X : A, y^Y : B \vdash \vec{s} : C}{\Gamma, \Delta \vdash \mathsf{let}_{(X,Y)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s} : C}\ (\text{LetPair})$$

$$\frac{\Gamma \vdash \vec{t} : \sharp(A \times B) \quad \Delta, x^X : \sharp A, y^Y : \sharp B \vdash \vec{s} : C}{\Gamma, \Delta \vdash \mathsf{let}_{(X,Y)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s} : \sharp C}\ (\text{LetTens})$$

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

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

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

$$\frac{\Gamma, x^X : B_X, y^X : B_X \vdash \vec{t} : B}{\Gamma, x^X : B_X \vdash \vec{t}\,[y := x] : B}\ (\text{Contr}) \qquad \frac{\Gamma \vdash \vec{t} : C}{\Gamma, x^X : B_X \vdash \vec{t} : C}\ (\text{Weak})$$

$$\frac{\Gamma \vdash \vec{t} : A \quad A \leq B}{\Gamma \vdash \vec{t} : B}\ (\text{Sub}) \qquad \frac{\Gamma \vdash \vec{t} : A \quad \vec{t} \equiv \vec{s}}{\Gamma \vdash \vec{s} : A}\ (\text{Equiv}) \qquad \frac{\Gamma \vdash \vec{t} : A}{\Gamma \vdash e^{i\theta}\vec{t} : A}\ (\text{Phase})$$

Table 3.5: *Some valid typing rules*

which case holds requires two queries to f. Deutsch's algorithm decides this with a single query by exploiting quantum superposition and interference.

Operationally, the oracle U_f acts as $U_f : |xy\rangle \mapsto |x\rangle \otimes |y \oplus f(x)\rangle$, where $\oplus$ is addition modulo 2. The textbook circuit prepares the state $|+-\rangle$, applies U_f, and then applies a Hadamard on the first qubit before measuring it. The outcome is $|0\rangle$ if f is constant and $|1\rangle$ if f is balanced.

We begin with a standard implementation of this algorithm in λ_B. We first encode the usual gates we will use:

$$\begin{aligned}
\mathbb{H} &:= \lambda x_{\mathbb{B}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|0\rangle \mapsto |+\rangle \mid |1\rangle \mapsto |-\rangle\},\\
\mathsf{NOT} &:= \lambda x_{\mathbb{B}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|0\rangle \mapsto |1\rangle \mid |1\rangle \mapsto |0\rangle\},\\
\mathsf{CNOT} &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|0\rangle \mapsto (|0\rangle\,, y) \mid |1\rangle \mapsto (|1\rangle\,, \mathsf{NOT}\ y)\}.
\end{aligned}$$

We model the four possible oracles U_f (two constant and two balanced):

$$\begin{aligned}
O_{\text{const}\,0} &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, (x, y), & O_{\text{id}} &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, \mathsf{CNOT}\ x\ y,\\
O_{\text{const}\,1} &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, (x, (\mathsf{NOT}\ y)), & O_{\text{flip}} &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, \mathsf{CNOT}\ x\ (\mathsf{NOT}\ y).
\end{aligned}$$

The standard Deutsch term prepares $|+-\rangle$, calls the oracle, then applies H on the first qubit and returns the (classical) first component:

$$\mathsf{Deutsch}_{\mathrm{std}} := \lambda f_{\mathcal{P}} \,.\, \mathsf{let}_{(\mathbb{B},\mathbb{B})}\ (x,y) = (f\,(\mathbb{H}\ |0\rangle)\,(\mathbb{H}\ |1\rangle))\ \mathsf{in}\ (\mathbb{H}\,x, y).$$

Each oracle above can be typed as $B_{\mathbb{B}} \Rightarrow B_{\mathbb{B}} \Rightarrow (B_{\mathbb{B}} \times B_{\mathbb{B}})$, but since the arguments we pass are $|+\rangle$ and $|-\rangle$ (superpositions), the overall judgement we can derive for the application uses $\sharp$:

$$\vdash \mathsf{Deutsch}_{\mathrm{std}} : (\sharp B_{\mathbb{B}} \Rightarrow \sharp B_{\mathbb{B}} \Rightarrow (\sharp B_{\mathbb{B}} \times \sharp B_{\mathbb{B}})) \Rightarrow \sharp(B_{\mathbb{B}} \times B_{\mathbb{B}}).$$

This type is correct but coarse: it only guarantees that the result is a unitary distribution of pairs of booleans. Operationally we know more: the first output is actually a basis bit ($|0\rangle$ or $|1\rangle$) encoding whether f is constant or balanced, and the second output can be ignored.

Then, we can consider a basis-aware implementation with tighter typing. The key observation is that the oracle is always called on the fixed state $|+-\rangle$. Thus it is natural to write the program in the $\mathbb{X} = \{|+\rangle, |-\rangle\}$ basis, letting the types track that we remain in a basis state at the oracle boundary.

We define the same gates in the $\mathbb{X}$ basis:

$$\begin{aligned}
\mathsf{Z}_{\mathbb{X}} &:= \lambda x_{\mathbb{X}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|+\rangle \mapsto |-\rangle \mid |-\rangle \mapsto |+\rangle\},\\
\mathsf{NOT}_{\mathbb{X}} &:= \lambda x_{\mathbb{X}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|+\rangle \mapsto |+\rangle \mid |-\rangle \mapsto -1\ |-\rangle\},\\
\mathsf{CNOT}_{\mathbb{X}} &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, \mathsf{case}\ y\ \mathsf{of}\{|+\rangle \mapsto (x, |+\rangle) \mid |-\rangle \mapsto (\mathsf{Z}_{\mathbb{X}} x, |-\rangle)\}.
\end{aligned}$$

We then rewrite the program and the four oracles in $\mathbb{X}$:

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

$$\begin{aligned}
O^{\mathbb{X}}_{\mathrm{const}\,0} &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, (x,y), & O^{\mathbb{X}}_{\mathrm{id}} &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, \mathsf{CNOT}_{\mathbb{X}}\ x\ y,\\
O^{\mathbb{X}}_{\mathrm{const}\,1} &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, (x, (\mathsf{NOT}_{\mathbb{X}}\ y)), & O^{\mathbb{X}}_{\mathrm{flip}} &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, \mathsf{CNOT}_{\mathbb{X}}\ x\ (\mathsf{NOT}_{\mathbb{X}}\ y).
\end{aligned}$$

Now every oracle has the tight type $B_{\mathbb{X}} \Rightarrow B_{\mathbb{X}} \Rightarrow (B_{\mathbb{X}} \times B_{\mathbb{X}})$, and the program itself can be typed as

$$\vdash \mathsf{Deutsch} : (B_{\mathbb{X}} \Rightarrow B_{\mathbb{X}} \Rightarrow (B_{\mathbb{X}} \times B_{\mathbb{X}})) \Rightarrow B_{\mathbb{B}}.$$

Intuitively, the oracle—when fed with $|+-\rangle$—produces a pair of basis states in $\mathbb{X}$ (up to a global phase). Hence we can treat its output classically: a single case on the first component suffices to return a classical bit in the computational basis, with no need for a $\sharp$-type on the result.

Both implementations are operationally equivalent: they compute a bit that decides whether f is constant or balanced. The difference lies in the *precision* of their typings. The standard version, expressed in $\mathbb{B}$ with explicit Hadamards, forces $\sharp$ on the oracle's interface and thus yields a result in $\sharp(B_{\mathbb{B}} \times B_{\mathbb{B}})$. The basis-aware version states, via types, that the oracle is used on a fixed $\mathbb{X}$-input and therefore returns an $\mathbb{X}$-basis pair; this lets us deterministically extract a $\mathbb{B}$-basis bit. In the typing derivation, no variable carries a $\sharp$-type: (i) we may safely ignore the second qubit, and (ii) the first qubit is guaranteed to be classical in $\mathbb{B}$, reflecting the determinism of Deutsch's algorithm.

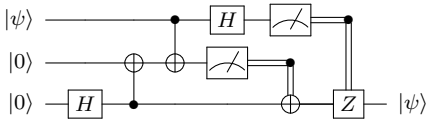
3.5.2 Quantum teleportation

We now turn to the *quantum teleportation protocol*, a cornerstone of quantum information theory that exemplifies the manipulation of entangled states and the transmission of quantum data through classical communication. Within the λ_B calculus, this protocol provides a natural setting to combine pattern matching, linear handling of qubits, and the *deferred-measurement principle*. Using the case constructor together with the expressive typing discipline introduced earlier, we can encode the teleportation process in a way that

remains both syntactically compact and semantically faithful to its quantum-mechanical counterpart.

The deferred-measurement principle states that any quantum circuit can delay its measurements without altering the final outcome. More precisely, any gate classically controlled by the result of a measurement is equivalent to another circuit where the control qubit remains unmeasured, acting coherently on all branches of the superposition. Although the λ_B calculus has no primitive operation for measurement, the case constructor allows us to simulate such classically controlled gates by branching on basis states.

A canonical example that exploits this principle is the *quantum teleportation protocol*. Two agents (Alice and Bob) share an entangled pair of qubits forming a Bell state. Using this shared entanglement and two bits of classical information, Alice can transmit an unknown quantum state $|\psi\rangle$ to Bob without physically sending the qubit itself. The standard circuit implementing the protocol is shown below:



The algorithm first creates the Bell state Φ^+ between the second and third qubits, then performs a Bell-basis measurement on the first and second qubits. Operationally, this measurement is implemented by applying a CNOT gate followed by a Hadamard on the first qubit (the adjoint of Bell-state preparation), and then measuring both qubits. Depending on the pair of classical outcomes, a Pauli correction (I, X, Z, or ZX) is applied to Bob's qubit to recover the original state $|\psi\rangle$.

We can simulate the behaviour of this circuit in the λ_B calculus by defining a term that, instead of performing measurements, explicitly describes the computation corresponding to each branch. A possible encoding is:

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

This term takes the input qubit $|\psi\rangle$ and pairs it with one half of an entangled Bell pair. The case construct then matches the first two qubits (the input and Alice's entangled qubit) against the Bell basis, and in each branch applies the appropriate correction to Bob's qubit to recover $|\psi\rangle$. The resulting term represents the same quantum transformation as the circuit above but expressed without any explicit measurement—instead, each branch encodes the coherent superposition of possible measurement outcomes.

The λ_B calculus allows us to abstract both the encoding and decoding steps in the Bell basis, exploiting the deferred-measurement principle in a type-safe way. Each branch of the case corresponds to a unitary transformation preserving linearity and orthogonality, and the overall term has type

$$\vdash \mathsf{Teleport} : \sharp B_{\mathbb{B}} \Rightarrow (\sharp B_{\mathsf{Bell}} \times \sharp B_{\mathbb{B}}).$$

This type reflects that the protocol operates on superpositions of basis states, producing a Bell-basis measurement outcome together with the recovered qubit. In this way, the λ_B calculus captures both the logical structure of teleportation and its deferred-measurement semantics within a single, uniform term language.

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

3.6 Conclusion

In this chapter we have explored a quantum-control λ-calculus equipped with the additional feature of allowing abstractions to be expressed relative to arbitrary bases, beyond the canonical one.

The central mechanism enabling this extension is the decoration of λ-abstractions and let-constructors with basis annotations, together with a modified substitution operation that governs how value distributions decompose across different bases. These additions do not increase the expressive power of the original calculus on which λ_B builds, yet they offer a novel perspective for reasoning about quantum programs and their behaviour under basis changes.

The reduction system coordinates computation through these extended syntactic constructs and substitutions. A key property is that evaluation commutes with the congruence relation, ensuring that interpreting a value distribution in a different basis does not affect the computational result. Consequently, it is meaningful to reason about terms modulo basis congruence.

The benefit of this design becomes clear in the realisability model. The inclusion of atomic types B_X enables a direct characterisation of abstractions representing unitary operators—our main semantic result, generalising the characterisation from [19]. Here, the use of basis types yields a simpler and more transparent proof.

The second major result is the validity of the typing rules presented in 3.5. By deriving these rules from the realisability interpretation, we ensure their soundness and obtain a principled foundation for a typed programming language based on the calculus.

Finally, we have illustrated the expressive advantages of the system through two canonical examples. In the case of *Deutsch's algorithm*, the use of basis-aware typing allows the result to be treated classically, reflecting the algorithm's determinism. In the case of *quantum teleportation*, we demonstrated how the case construct can simulate gates controlled by Bell-basis measurements, effectively capturing the deferred-measurement principle within the calculus.

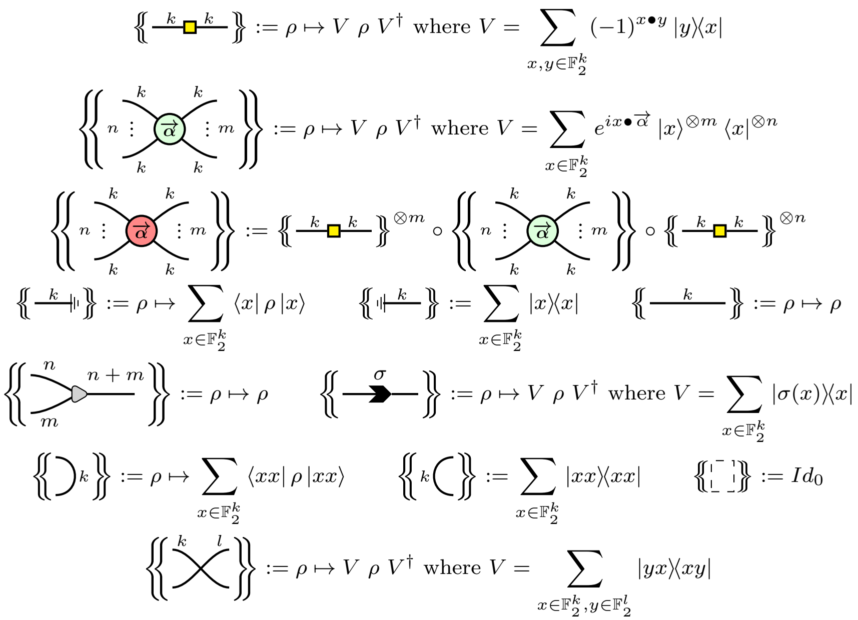
Appendices

.1 Semantics of the SZX calculus

We reproduce below the standard interpretation of $SZX_{\perp}$ diagrams as density matrices and completely positive maps [9, 7], modulo scalars.

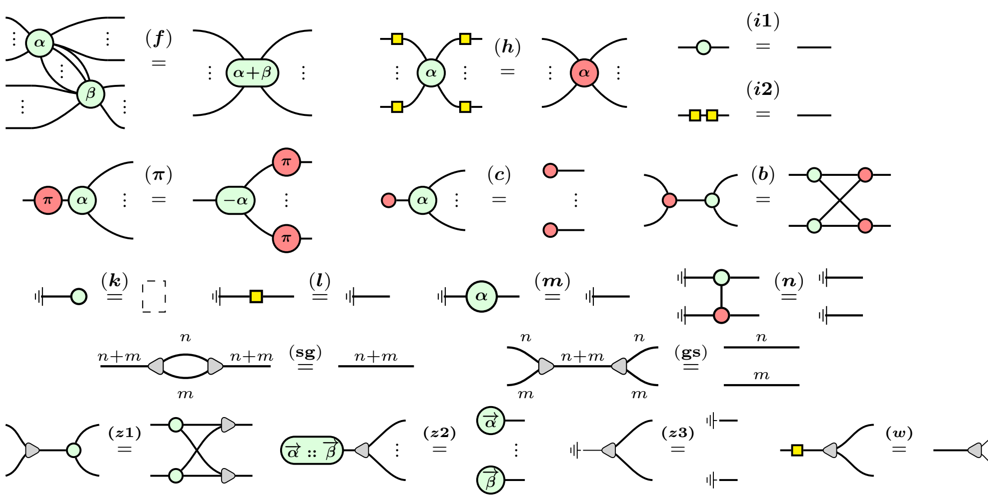
Let $D_n \subseteq \mathbb{C}^{2^n \times 2^n}$ be the set of n-qubit density matrices. We define the functor $\{\cdot\} : ZX_{\perp} \to \mathbf{CPM}(\mathbf{Qubit})$ which associates to any diagram $\mathscr{D} : n \to m$ a completely positive map $\{\mathscr{D}\} : D_n \to D_m$, inductively as follows.

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

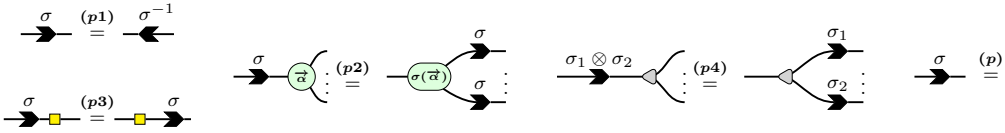


where $\forall u, v \in \mathbb{R}^n, u \bullet v = \sum_{i=1}^{m} u_i v_i$.

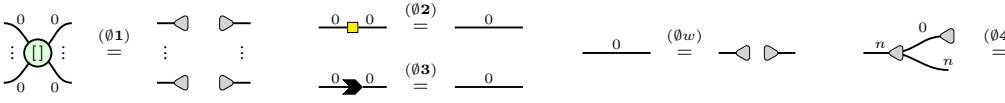
The $SZX_{\perp}$ calculus defines a set of rewrite rules, shown below.



Additionally, for the arrows restricted to permutations of wires we have the following rules [7]:



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 6.

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 7.

Additionally, we define useful macros based on these functions on Table 8. 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)
        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
```

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

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

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

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

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

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

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

Table 6: *Rewrite system for λ_D.*

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

Table 7: *Reductions pertaining to the primitives.*

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

Table 8: *Function macros.*

```
    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'
```

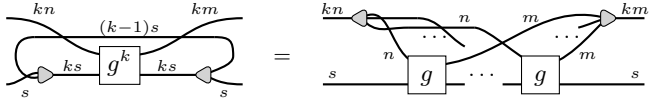
.4 QFT algorithm in Quipper code

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.

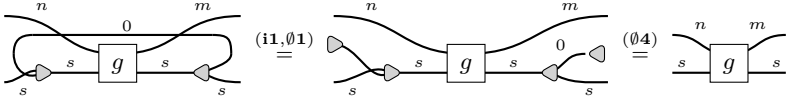
```
module Qft where
import "/dpq/Prelude.dpq"

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

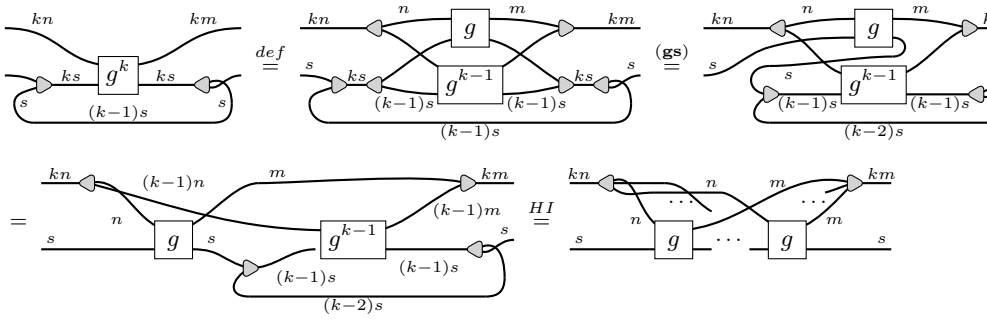
Lemma 11. *(1) Let* $g : 1_n \otimes 1_s \to 1_m \otimes 1_s$ *and* $k \geq 1$*, then*



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

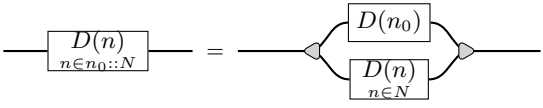


If $k > 1$,



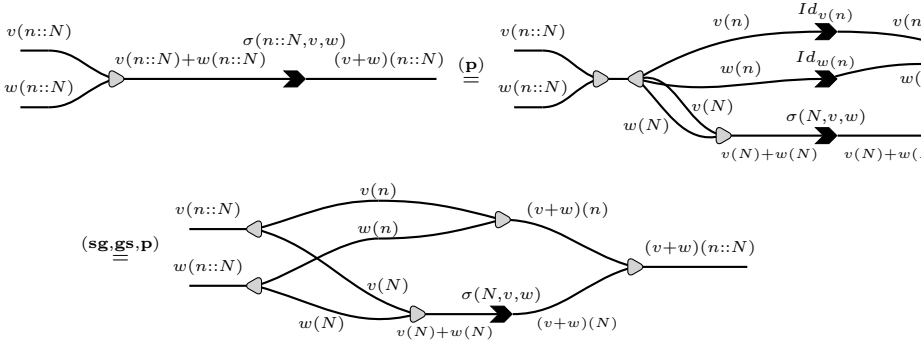
□

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



Proof. By induction on the term construction

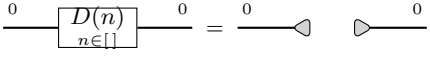
- If $\mathscr{D}$ is a gather,



- 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 $\Phi \vdash n : \mathtt{Nat}$, 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 $\Phi \vdash M \square N : \mathtt{Nat} \Phi, \Gamma, \Delta \vdash M :: N \ : \mathtt{Vec}\ (n+1)\ A$, then $\lfloor M \square N \rfloor_\Phi = |\Phi| \mapsto \lfloor M \rfloor_\Phi (|\Phi|) \square \lfloor N \rfloor_\Phi (\ |\Phi|\ )$. By inductive hypothesis, $\lfloor M \rfloor_\Phi (|\Phi|), \lfloor M \rfloor_\Phi (|\Phi|) \in \mathbb{N}$. Then, $|\Phi| \mapsto \lfloor M \rfloor_\Phi (|\Phi|) \square \lfloor N \rfloor_\Phi (|\Phi|) \in \bigtimes_{x:P \in \Phi} \lfloor P \rfloor \to \mathbb{N}$.
- 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 $x \mapsto \lfloor M \rfloor_\Phi (x, \Phi) \in \lfloor x : \mathtt{Nat} \to B[x] \rfloor$. 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 $\Phi \vdash \mathtt{VNil}^A : \mathtt{Vec}\ 0\ A$, then $\left\lfloor \mathtt{VNil}^P \right\rfloor_\Phi = |\Phi| \mapsto [] \in \bigtimes_{x:P \in \Phi} \lfloor P \rfloor \to \mathbb{N}^0$.
- If $\Phi, \Gamma, \Delta \vdash M :: N \ : \mathtt{Vec}\ (n+1)\ A$, 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 $\Phi, \Gamma \vdash \mathtt{ifz}\ L\ \mathtt{then}\ M\ \mathtt{else}\ N : A$, then
  $\lfloor \mathtt{ifz}\ L\ \mathtt{then}\ M\ \mathtt{else}\ N \rfloor_\Phi = |\Phi| \mapsto \begin{cases} \lfloor M \rfloor_\Phi (|\Phi|) \text{ if } \lfloor L \rfloor_\Phi (|\Phi|) = 0 \\ \lfloor N \rfloor_\Phi (|\Phi|) \text{ otherwise} \end{cases}$.
  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 $\lfloor \mathtt{ifz}\ L\ \mathtt{then}\ M\ \mathtt{else}\ N \rfloor_\Phi \in \bigtimes_{x:P \in \Phi} \lfloor P \rfloor \to \lfloor A \rfloor$.
- If $\Phi \vdash \mathtt{for}\ k\ \mathtt{in}\ N\ \mathtt{do}\ M : \mathtt{Vec}\ n\ A$, then $\lfloor \mathtt{for}\ k\ \mathtt{in}\ V\ \mathtt{do}\ M \rfloor_\Phi = |\Phi| \mapsto \bigtimes_{k \in \lfloor N \rfloor_\Phi (|\Phi|)} \lfloor M \rfloor_{k:\mathtt{Nat}\Phi}$ $|\Phi|)$. 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 $M = x$, $M = n$, $M = \texttt{VNil}^{\texttt{Nat}}$, $M = \lambda' x.M'$, $M = M_1 :: M_2$ or $M = \texttt{range}$ then M is in normal form and it does not reduce.
- If $M = M_1 \Box M_2$ we have three cases:
  - If $M \to M_1 \Box N$ with $M_2 \to N$, then:
$$\begin{aligned}\lfloor M_1 \Box M_2 \rfloor_\Phi &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (|\Phi|) \Box \lfloor M_2 \rfloor_\Phi (|\Phi|) \\ &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (|\Phi|) \Box \lfloor N \rfloor_\Phi (|\Phi|) \\ &= \lfloor M_1 \Box N \rfloor_\Phi\end{aligned}$$
  - If $M \to N \Box V$ with $M_1 \to N$, then:
$$\begin{aligned}\lfloor M_1 \Box V \rfloor_\Phi &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (|\Phi|) \Box \lfloor V \rfloor_\Phi (|\Phi|) \\ &= |\Phi| \mapsto \lfloor N \rfloor_\Phi (|\Phi|) \Box \lfloor V \rfloor_\Phi (|\Phi|) \\ &= \lfloor N \Box V \rfloor_\Phi\end{aligned}$$
  - If $M \to n$ with $M_i = n_i \in \mathbb{N}$ and $n = n_1 \Box n_2$, then:
$$\begin{aligned}\lfloor n_1 \Box n_2 \rfloor_\Phi &= |\Phi| \mapsto \lfloor n_1 \rfloor_\Phi (|\Phi|) \Box \lfloor n_2 \rfloor_\Phi (|\Phi|) \\ &= |\Phi| \mapsto n_1 \Box n_2 \\ &= |\Phi| \mapsto n \\ &= \lfloor n \rfloor_\Phi\end{aligned}$$
- 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 $M \to N \ @ \ V$ with $M_1 \to N$, then:
$$\begin{aligned}\lfloor M_1 \ @ \ V \rfloor_\Phi &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (\lfloor V \rfloor_\Phi (|\Phi|), \Phi) \\ &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (\lfloor V \rfloor_\Phi (|\Phi|), \Phi) \\ &= \lfloor N \ @ V \rfloor_\Phi\end{aligned}$$
  - If $M \to M'[V/x]$ with $M_1 = \lambda' x.M'$ and $M_2 = V$, then:
$$\begin{aligned}\left\lfloor (\lambda' x.M) @ V \right\rfloor_\Phi &= |\Phi| \mapsto \left\lfloor \lambda' x.M \right\rfloor_\Phi (\lfloor V \rfloor_\Phi (|\Phi|), \Phi) \\ &= |\Phi| \mapsto (x, |\Phi| \mapsto \lfloor M \rfloor_{x,\Phi} (x, |\Phi|))(\lfloor V \rfloor_\Phi (|\Phi|), \Phi) \\ &= |\Phi| \mapsto \lfloor M[V/X] \rfloor_\Phi (|\Phi|) \\ &= \lfloor M[V/X] \rfloor_\Phi\end{aligned}$$
- 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:

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

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

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

- 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:

$$\left\lfloor \texttt{for}\ k\ \texttt{in}\ M'\ \texttt{do}\ R \right\rfloor_\Phi = |\Phi| \mapsto \bigtimes_{k \in \lfloor M' \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)$$
$$= |\Phi| \mapsto \bigtimes_{k \in \lfloor N \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)$$
$$= \lfloor \texttt{for}\ k\ \texttt{in}\ N\ \texttt{do}\ R \rfloor_\Phi$$

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

$$\lfloor \texttt{for}\ k\ \texttt{in}\ M_1\ ::\ M_2\ \texttt{do}\ R \rfloor_\Phi = |\Phi| \mapsto \bigtimes_{k \in \lfloor M_1\ ::\ M_2 \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)$$
$$= |\Phi| \mapsto \bigtimes_{k \in \lfloor M_1 \rfloor_\Phi (|\Phi|) \times \lfloor M_2 \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)$$
$$= |\Phi| \mapsto \lfloor R \rfloor_{k,\Phi} (\lfloor M_1 \rfloor_\Phi (|\Phi|), |\Phi|) \times \bigtimes_{k \in \lfloor M_2 \rfloor_\Phi (|\Phi|} \lfloor R \rfloor_{k,\Phi} (k,$$
$$= \lfloor R[M_1/k] \rfloor_\Phi \times \lfloor \texttt{for}\ k\ \texttt{in}\ M_2\ \texttt{do}\ R \rfloor_\Phi$$
$$= \lfloor R[M_1/k]\ ::\ \texttt{for}\ k\ \texttt{in}\ M_2\ \texttt{do}\ R \rfloor_\Phi$$

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

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

$$= |\Phi| \mapsto \bigtimes_{k \in []} \lfloor R \rfloor_\Phi (k, |\Phi|)$$

$$= |\Phi| \mapsto []$$

$$= \left\lfloor \texttt{VNil}^{\texttt{Nat}} \right\rfloor_\Phi$$

- 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:

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

  - 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:

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

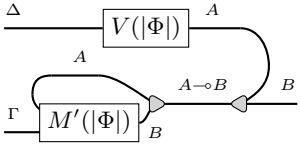
□

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 $[\![M]\!]_{\Phi,\Gamma} = [\![N]\!]_{\Phi,\Gamma}$.*

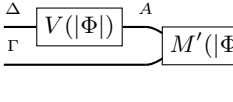
Proof. By case analysis on the reductions of translatable terms.

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

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



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



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

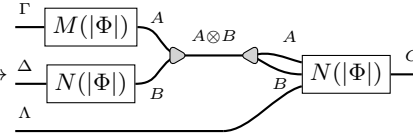
- If $M = (\lambda' x^A.M')@V$ and $N = M'[V/x]$,

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

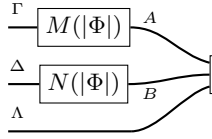
$$= |\Phi| \mapsto \overset{\Gamma}{\text{———}}\boxed{\twoheadrightarrow (M'_\Phi(\twoheadrightarrow V_\Phi(|\Phi|), |\Phi|)}\overset{B}{\text{———}} = \left[\!\left[M'[V/x]\right]\!\right]_{\Phi,\Gamma}$$

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

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



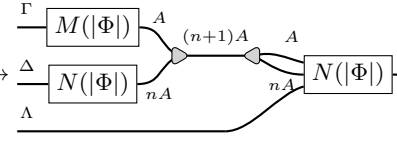
$$\overset{(\mathbf{gs})}{=} |\Phi| \mapsto$$



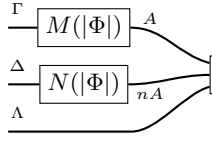
$$= \left[\!\left[M'[V_1/x][V_2/y]\right]\!\right]_{\Phi,\Delta,\Gamma,\Lambda}$$

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

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



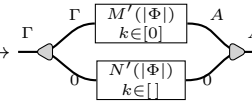
$$\overset{(\mathbf{gs})}{=} |\Phi| \mapsto$$



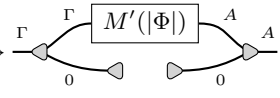
$$= \left[\!\left[M'[V_1/x][V_2/y]\right]\!\right]_{\Phi,\Delta,\Gamma,\Lambda}$$

- If $M = \texttt{ifz}\ L\ \texttt{then}\ M'\ \texttt{else}\ N'$,
  - if $L = 0$ and $N = M'$, $\lfloor L \rfloor_\Phi = 0$ and

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



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



$$\overset{(\emptyset\mathbf{4})}{=} |\Phi| \mapsto \overset{\Gamma}{\text{———}}\boxed{M'(|\Phi|)}\overset{A}{\text{———}} = \left[\!\left[M'\right]\!\right]_{\Phi,\Gamma}$$

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

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

$$\left[\!\left[\texttt{VNil};_v M'\right]\!\right]_{\Phi,\Gamma} = |\Phi| \mapsto \overset{\Gamma}{\text{———}}\boxed{M'(|\Phi|)}\overset{A}{\text{———}} = \left[\!\left[M'\right]\!\right]_{\Phi,\Gamma}$$

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

$$\left[\!\left[\star; M'\right]\!\right]_{\Phi,\Gamma} = |\Phi| \mapsto \overset{\Gamma}{\text{———}}\boxed{M'(|\Phi|)}\overset{A}{\text{———}} = \left[\!\left[M'\right]\!\right]_{\Phi,\Gamma}$$

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

$$\left[\!\left[\texttt{for}\ k\ \texttt{in}\ V :: M'\ \texttt{do}\ N'\right]\!\right]_{\Phi,\Gamma^{\otimes n}} = |\Phi| \mapsto \overset{\Gamma^{\otimes n}}{\text{---}}\boxed{\underset{k\in\twoheadrightarrow V::M'(|\Phi|)}{N'(k,|\Phi|)}}\overset{A^{\otimes n}}{\text{---}}$$

$$\overset{\text{Lemma 2}}{=} |\Phi| \mapsto \overset{\Gamma^{\otimes n}}{\text{---}}\!\!<\!\!\begin{cases}\Gamma & \boxed{N'(\twoheadrightarrow V(|\Phi|),|\Phi|)} & A \\ \Gamma^{\otimes n-1} & \boxed{\underset{k\in\twoheadrightarrow M'(|\Phi|)}{N'(k,|\Phi|)}} & A^{\otimes n-1}\end{cases}\!\!>\!\!\overset{A^{\otimes n}}{\text{---}} = \left[\!\left[N'[k/V] :: \texttt{for}\ k\ \texttt{in}\ M'\ \texttt{do}\ N'\right]\!\right]_{\Phi,\Gamma^{\otimes n}}$$

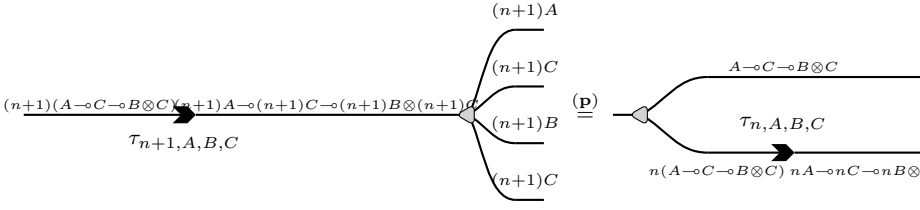
- If $M = \texttt{for}\ k\ \texttt{in}\ \texttt{VNil}\ \texttt{do}\ N'$ and $N = \texttt{VNil}$,

$$\left[\!\left[\texttt{for}\ k\ \texttt{in}\ \texttt{VNil}\ \texttt{do}\ N'\right]\!\right]_{\Phi} = |\Phi| \mapsto \overset{0}{\text{---}}\boxed{\underset{k\in[\,]}{N'(k,|\Phi|)}}\overset{0}{\text{---}}$$

$$\overset{\textit{Lemma 3}}{=} |\Phi| \mapsto \overset{0}{\text{---}}\!\!\triangleleft \qquad \triangleright\!\!\overset{0}{\text{---}} = |\Phi| \mapsto [\,] = \left[\!\left[\texttt{VNil}\right]\!\right]_{\Phi}$$

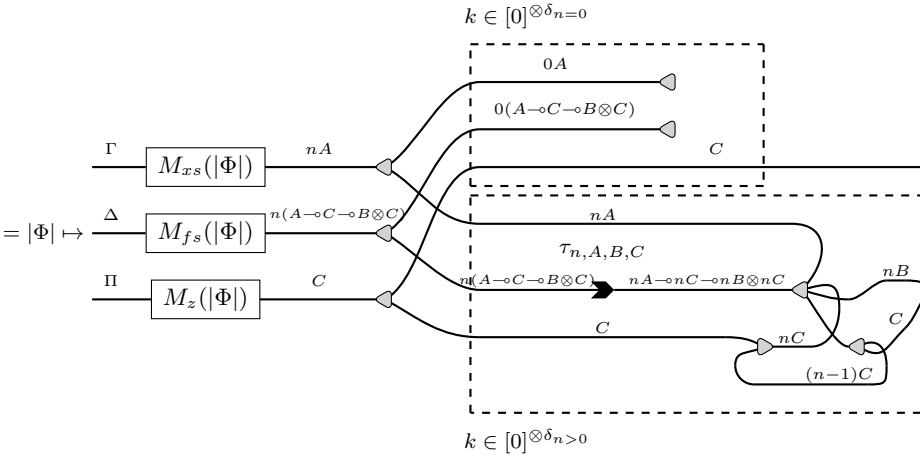
- If $M = \texttt{accuMap}_{A,B,C}\ N'\ M_{xs}\ M_{fs}\ M_z$ and $N = \texttt{ifz}\ N'\ \texttt{then}\ xs\ ;_v\ fs\ ;_v\ \texttt{VNil} \otimes M_z\ \texttt{else}\ \texttt{let}\ x :: xs' = M_{xs}\ \texttt{in}\ \texttt{let}\ f :: fs' = M_{fs}\ \texttt{in}\ \texttt{let}\ y \otimes z' = f\ x\ z\ \texttt{in}\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

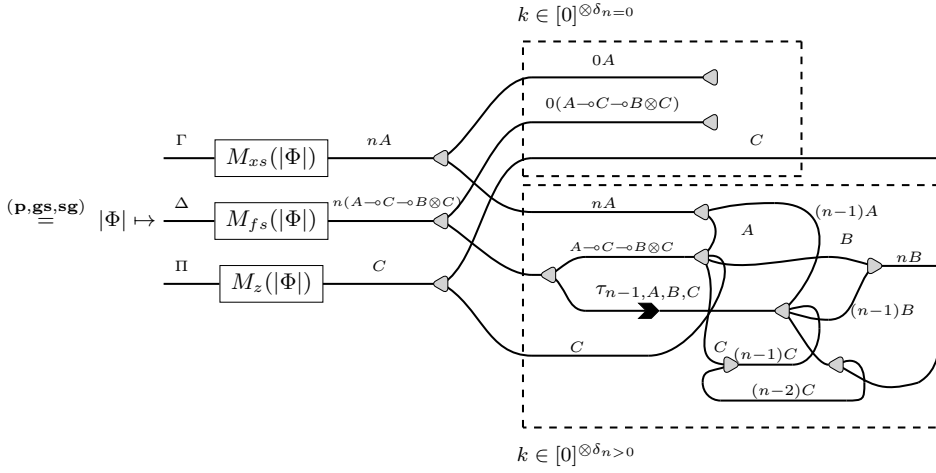
  Notice that, by definition of $\tau_{n,A,B,C}$,



  Therefore,

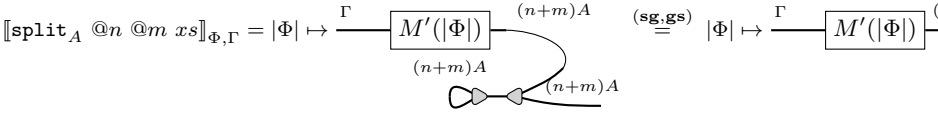
$$\left[\!\left[\texttt{accuMap}_{A,B,C}\ N'\ M_{xs}\ M_{fs}\ M_z\right]\!\right]_{\Phi,\Gamma,\Delta,\Pi}$$

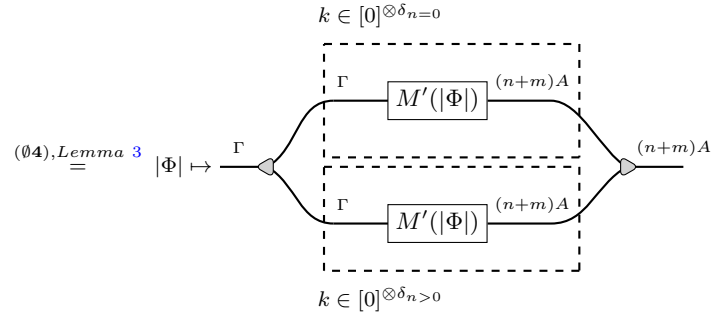


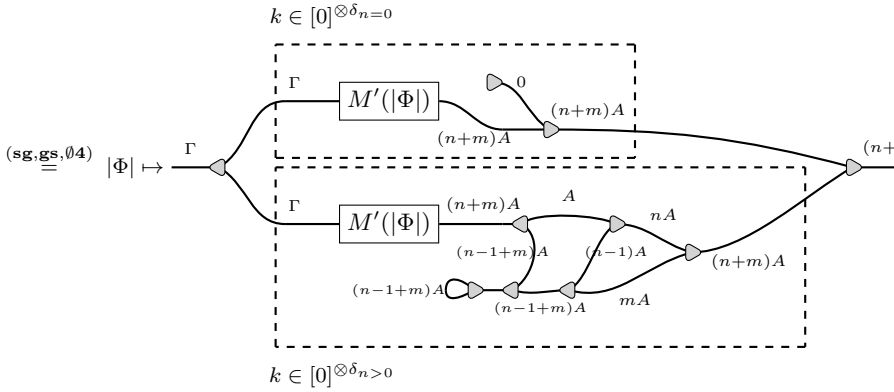


$= [\![\texttt{ifz}\ N' \ \texttt{then}\ xs\ ;_v\ fs\ ;_v\ \texttt{VNil} \otimes M_z\ \texttt{else let}\ x :: xs' = M_{xs}\ \texttt{in let}\ f :: fs' = M_{fs}\ \texttt{in let}\ y \otimes z' = f\ x\ z\ \texttt{in let}\ ys \otimes z'' = \texttt{accuMap}\ @(N'-1)\ xs'\ fs'\ z'\ \texttt{in}\ (y :: ys) \otimes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

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



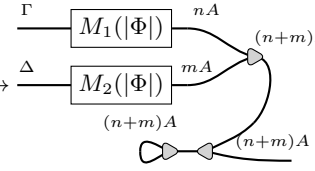


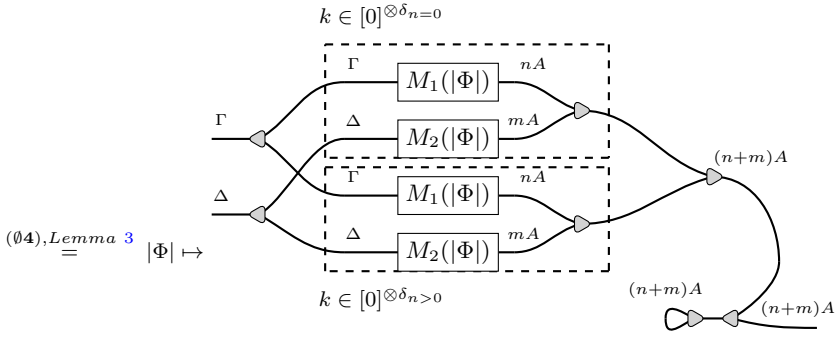


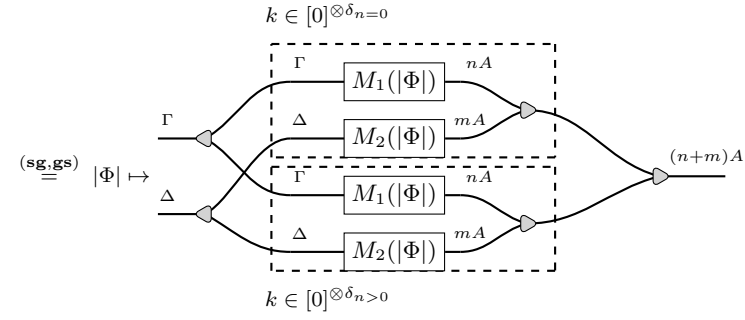
$= [\![\texttt{ifz}\ n\ \texttt{then VNil} \otimes xs\ \texttt{else let}\ y :: xs' = xs\ \texttt{in let}\ ys_1 \otimes ys_2 = \texttt{split@}(n-1)\ @m\ xs'\ \texttt{in}\ (y :: ys_1) \otimes ys_2]\!]_{\Phi,\Gamma}$

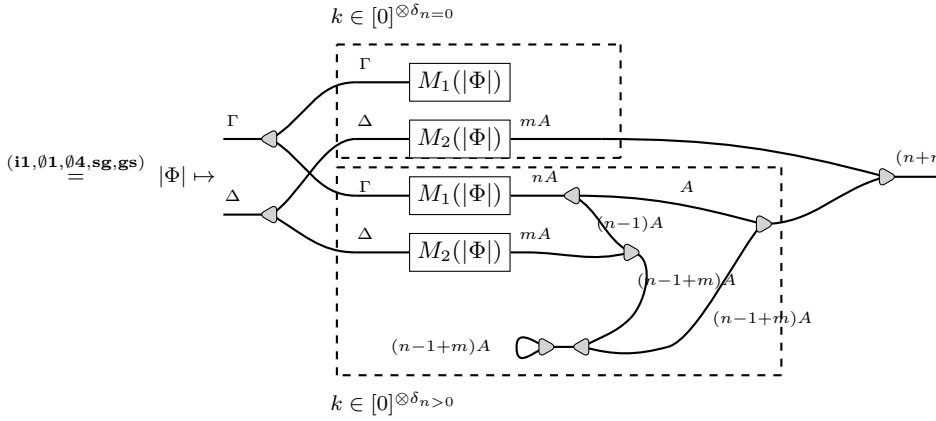
- If $M = \texttt{append}_A\ @N_1\ @N_2\ M_1\ M_2$ and $N = \texttt{ifz}\ N_1\ \texttt{then}\ M_1\ ;_v\ M_2\ \texttt{else let}\ x :: xs' = M_1\ \texttt{in}\ x :: (\texttt{append}\ @(N_1 - 1)\ @N_2\ M_1\ M_2)$. Let $n = \lfloor N_1 \rfloor_\Phi (|\Phi|)$ and $m = \lfloor N_2 \rfloor_\Phi (|\Phi|)$.

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







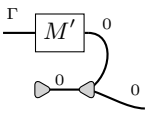


$\overset{(\mathbf{i1},\emptyset\mathbf{1},\emptyset\mathbf{4},\mathbf{sg},\mathbf{gs})}{=} |\Phi| \mapsto$

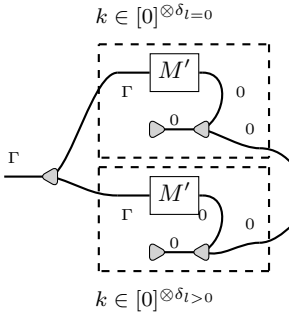
$= [\![\texttt{ifz}\ N_1\ \texttt{then}\ M_1\ ;_v\ M_2\ \texttt{else let}\ x :: xs' = M_1\ \texttt{in}\ x :: (\texttt{append}\ @(N_1 - 1)\ @N_2\ M_1\ M_2]\!]_{\Phi,\Gamma}$

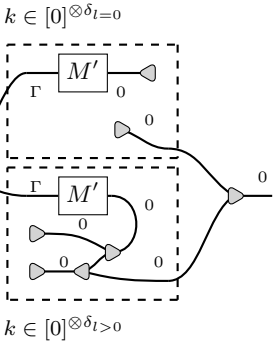
- If $M = \texttt{drop}\ @N'\ M'$ and $N = \texttt{ifz}\ N'\ \texttt{then}\ M'\ ;\ \star\ \texttt{else let}\ x :: xs' = M'\ \texttt{in}\ x\ ;\ \texttt{drop}\ @(N'-1)\ xs'$. Let $l = \lfloor N' \rfloor_\Phi\,(|\Phi|)$,

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



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





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

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

- 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.

□

.6 Omitted proofs from Section 3.2

Theorem (Restatement of 1)**.** *If B is an n-dimensional basis, then every n-dimensional qubit has a unique decomposition over B.*

Proof. Let $\vec{b_i}$ be the basis vectors of B. Suppose $\sum_{i=1}^n \alpha_i \vec{b_i}$ and $\sum_{i=1}^n \beta_i \vec{b_i}$ are two decompositions of $\vec{v}$ over B. Then

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

By linear independence, $\alpha_i = \beta_i$ for all i. □

Corollary (Restatement of 1)**.** *If $\vec{v} \equiv \vec{w}$, then they share the same decomposition over any basis B.*

Proof. Since $\vec{v} - \vec{w} \equiv \vec{v} - \vec{v} \equiv \vec{w} - \vec{w}$, the same argument as in 1 shows that $\vec{v}$ and $\vec{w}$ have the same decomposition over B. □

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

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

Proof. Let $B \neq \mathcal{P}$ and $\vec{v} \equiv \sum_{j=1}^n \beta_j \vec{b_j}$ with each $\vec{b_j} \in B$. Then

$$\begin{aligned}\left(\sum_i \alpha_i \vec{t_i}\right) \langle \vec{v}/x \rangle_B &= \sum_{j=1}^{n} \beta_j \left(\sum_{i=1}^{n} \alpha_i t_i\right)[\vec{b_j}/x] \\ &\equiv \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{n} \beta_j t_i [\vec{b_j}/x]\right) = \sum_{i=1}^{n} \alpha_i \vec{t_i} \langle \vec{v}/x \rangle_B.\end{aligned}$$

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

Lemma (Restatement of 9)**.** *For value distributions $\vec{v}, \vec{w}$, a term distribution $\vec{t}$, and an orthonormal basis B such that both $\langle \vec{v}/x \rangle_B$ 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 1, both can be written as $\vec{v} \equiv \vec{w} \equiv \sum_{i=1}^n \alpha_i \vec{b_i}$ with $\vec{b_i} \in B$. Hence

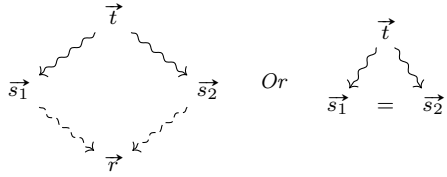
$$\vec{t}\langle \vec{v}/x \rangle_B = \sum_{i=1}^{n} \alpha_i \, \vec{t}\,[\vec{b_i}/x] = \vec{t}\langle \vec{w}/x \rangle_B.$$

□

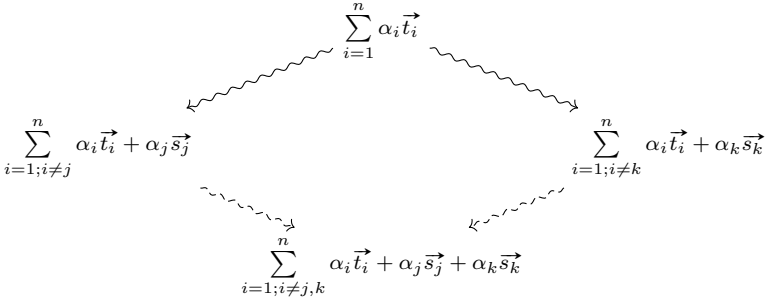
.7 Omitted proofs from Section 3.3

Lemma 19 (Weak diamond property for $\rightsquigarrow$)**.** *Let $\vec{t}, \vec{s_1}, \vec{s_2}$ term distributions such that $\vec{t} \rightsquigarrow s_1$ and $\vec{t} \rightsquigarrow \vec{s_2}$. Then, either there exists a term distribution $\vec{r}$ such that $\vec{s_1} \rightsquigarrow \vec{r}$*

and $\vec{s_2} \rightsquigarrow \vec{r}$*. Or,* $\vec{s_1} = \vec{s_2}$*. Diagrammatically:*

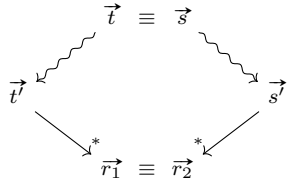


Proof. The proof follows from the fact that the $\rightsquigarrow$ reduction is deterministic over basic values. And, in case of term distributions, we only need to match the reduction on the corresponding sub-terms. Let $\vec{t} = \sum_{i=1}^{n} \alpha_i \vec{t_i}$, $\vec{s_1} = \sum_{i=1;i\neq j}^{n} \alpha_i \vec{t_i} + \alpha_j \vec{s_j}$, and $\vec{s_2} = \sum_{i=1;i\neq k}^{n} \alpha_i \vec{t_i} + \alpha_k \vec{s_k}$. Where $\vec{t_j} \rightsquigarrow \vec{s_j}$ and $\vec{t_k} \rightsquigarrow \vec{s_k}$. If $j = k$ we are done, so we consider the case where $j \neq k$. Diagrammatically:



□

Theorem (Restatement of 2)**.** *Let* $\vec{t}$ *and* $\vec{s}$ *be closed term distributions with* $\vec{t} \equiv \vec{s}$*. If* $\vec{t} \rightsquigarrow \vec{t'}$ *and* $\vec{s} \rightsquigarrow \vec{s'}$*, 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:*



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 19 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 19.

$\alpha(\beta\,\vec{t}) \equiv \delta\,\vec{t}$**:** This case follows from Lemma 19.

$\vec{t_1} + \vec{t_2} \equiv \vec{t_2} + \vec{t_1}$**:** This case follows from Lemma 19. 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 19. 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 19.

$\alpha(\vec{t_1} + \vec{t_2}) \equiv \alpha\vec{t_1} + \alpha\vec{t_2}$**:** 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 19.

$\vec{t}\,(\alpha\,\vec{s}) \equiv \alpha(\vec{t}\,\vec{s})$**:** 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 19.

If $\vec{t} = (\lambda x_B \,.\, \vec{t_1})$ and $\vec{s} = \vec{v}$ and $\vec{t_1}\langle\vec{v}/x\rangle_B$, is defined then (we consider the case $B \neq \mathcal{P}$):

$$(\lambda x_B \,.\, \vec{t_i})(\alpha\,\vec{v}) \rightsquigarrow \vec{t_1}\langle\alpha\,\vec{v}/x\rangle_B$$
$$= \sum_{i=1}^{n} \alpha\beta_i \vec{t_1}[\vec{b_i}/x] \quad \text{with } \vec{v} \equiv \sum_{i=1}^{n} \beta_i \vec{b_i} \text{ with } \vec{b_i} \in B$$

On the other side:

$$\alpha((\lambda x_B \,.\, \vec{t_1})\,\vec{v}) \rightsquigarrow \alpha(\vec{t_1}\langle\vec{v}/x\rangle_B)$$
$$= \alpha(\sum_{i=1}^{n} \beta_i \vec{t_1}[\vec{b_i}/x]) \quad \text{with } \vec{v} \equiv \sum_{i=1}^{n} \beta_i \vec{b_i} \text{ with } \vec{b_i} \in B$$

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 19. There are no other possible redexes since the abstraction must be a basic 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 19. There are no other possible redexes since the abstraction must be a basic 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 19.

If $\vec{t} = (\lambda x_B \,.\, \vec{t_1})$, $\vec{s} = \vec{v}$, and $\vec{r} = \vec{w}$ with $\vec{t_1}\langle\vec{v}/x\rangle_B$ and $\vec{t_1}\langle\vec{w}/x\rangle_B$ defined then (we consider the case where $B \neq \mathcal{P}$):

$$(\lambda x_B \,.\, \vec{t_1})(\vec{v} + \vec{w}) \rightsquigarrow \vec{t_1}\langle\vec{v} + \vec{w}/x\rangle_B$$
$$= \sum_{i=1}^{n} (\alpha_i + \beta_i)\vec{t_1}[\vec{b_i}/x]$$
$$\text{where } \vec{v} \equiv \sum_{i=1}^{n} \alpha_i \vec{b_i}, \vec{w} \equiv \sum_{i=1}^{n} \beta_i \vec{b_i} \text{ with } \vec{b_i} \in B$$

On the other side:

$$(\lambda x_B \,.\, \vec{t_1})\,\vec{v} + (\lambda x_B \,.\, \vec{t_1})\vec{w} \rightsquigarrow \vec{t_1}\langle\vec{v}/x\rangle_B + (\lambda x_B \,.\, \vec{t_1})\vec{w}$$

$$\rightsquigarrow \vec{t_1}\langle\vec{v}/x\rangle_B + \vec{t_1}\langle\vec{w}/x\rangle_B$$
$$= \sum_{i=1}^{n}(\alpha_i)\vec{t_1}[\vec{b_i}/x] + = \sum_{i=1}^{n}(\alpha_i)\vec{t_1}[\vec{b_i}/x]$$
$$\text{where } \vec{v} \equiv \sum_{i=1}^{n}\alpha_i\vec{b_i}, \vec{w} \equiv \sum_{i=1}^{n}\beta_i\vec{b_i} \text{ with } \vec{b_i} \in B$$

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 19.

If $\vec{t} = \vec{v}$ and $\vec{s}\langle\vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2}$ is defined, then (we consider $B_1, B_2 \neq \mathcal{P}$):

$$\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = (\alpha\,\vec{v})\ \mathsf{in}\ \vec{s} \rightsquigarrow \vec{s}\langle\alpha\,\vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2}$$
$$= \sum_{i=1}^{n}\alpha\beta_i\,\vec{s}[\vec{v_i}/x_1][\vec{w_i}/x_2]$$
$$\text{where: } \vec{v} \equiv \sum_{i=1}^{n}\beta_i(\vec{v_i},\vec{w_i}) \text{ with } \vec{v_i} \in B_1, \vec{w_i} \in B_2$$

On the other side;

$$\alpha(\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = \vec{v}\ \mathsf{in}\ \vec{s}) \rightsquigarrow \alpha(\vec{s}\langle\vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2})$$
$$= \alpha\sum_{i=1}^{n}\beta_i\,\vec{s}[\vec{v_i}/x_1][\vec{w_i}/x_2]$$
$$\text{where: } \vec{v} \equiv \sum_{i=1}^{n}\beta_i(\vec{v_i},\vec{w_i}) \text{ with } \vec{v_i} \in B_1, \vec{w_i} \in B_2$$

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 19.

If $\vec{t} = \vec{v}$ and $\vec{s} = \vec{w}$ with $\vec{r}\langle\vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2}$ and $\vec{r}\langle\vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2}$ defined, then (we consider $B_1, B_2 \neq \mathcal{P}$):

$$\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = \vec{v} + \vec{w}\ \mathsf{in}\ \vec{r} \rightsquigarrow \vec{r}\langle\vec{v} + \vec{w}/x_1 \otimes x_2\rangle_{B_1\otimes B_2}$$
$$= \sum_{i=1}^{n}(\alpha_i + \beta_i)\,\vec{r}[\vec{v_i}/x_1][\vec{w_i}/x_2]$$
$$\text{where: } \vec{v} \equiv \sum_{i=1}^{n}\alpha_i(\vec{v_i},\vec{w_i}) \text{ with } \vec{v_i} \in B_1, \vec{w_i} \in B_2$$
$$\text{and: } \vec{w} \equiv \sum_{i=1}^{n}\beta_i(\vec{v_i},\vec{w_i}) \text{ with } \vec{v_i} \in B_1, \vec{w_i} \in B_2$$

On the other side:

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

where: $\overrightarrow{v} \equiv \sum_{i=1}^{n} \alpha_i(\overrightarrow{v_i}, \overrightarrow{w_i})$ with $\overrightarrow{v_i} \in B_1, \overrightarrow{w_i} \in B_2$

and: $\overrightarrow{w} \equiv \sum_{i=1}^{n} \beta_i(\overrightarrow{v_i}, \overrightarrow{w_i})$ with $\overrightarrow{v_i} \in B_1, \overrightarrow{w_i} \in B_2$

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

$$
\begin{aligned}
&\mathsf{case}\ \alpha\,\overrightarrow{t}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{s_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{s_n}\} \equiv \\
&\alpha(\mathsf{case}\ \overrightarrow{t}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{s_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{s_n}\})
\end{aligned}
\quad :
$$

If $\alpha = 0$, then there is no reduction possible on the right hand-side. If there are internal reductions on $\overrightarrow{t}$, then we match on both sides of the equivalence with Lemma 19.
If $\overrightarrow{t} = \overrightarrow{v} \equiv \sum_{i=1}^{n} \beta_i \overrightarrow{v_i}$. Then:

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

On the other side:

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

And we have that both terms are equivalent.

$$
\begin{aligned}
&\mathsf{case}\ (\overrightarrow{t} + \overrightarrow{s})\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{r_n}\} \equiv \\
&\mathsf{case}\ \overrightarrow{t}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{r_n}\} + \\
&\mathsf{case}\ \overrightarrow{s}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{r_n}\}
\end{aligned}
\quad :
$$

If there is an internal reduction on either $\overrightarrow{t}$ or $\overrightarrow{s}$, then we match the reductions on both sides of the equivalence with Lemma 19.
If $\overrightarrow{t} = \overrightarrow{v} \equiv \sum_{i=1}^{n} \alpha_i \overrightarrow{v_i}$, and $\overrightarrow{s} = \overrightarrow{w} \equiv \sum_{i=1}^{n} \beta_i \overrightarrow{v_i}$. Then:

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

On the other side:

$$\mathsf{case}\ \overrightarrow{t}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{r_n}\} + \mathsf{case}\ \overrightarrow{s}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{r_n}\}$$

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

$$\rightsquigarrow \sum_{i=1}^{n} \alpha_i \vec{r_i} + \sum_{i=1}^{n} \beta_i \vec{r_i}$$

And we have that both terms are equivalent.

□

.8 Omitted proofs from Section 3.4

Theorem (Restatement of 3)**.** *The interpretation of a type $\sharp A$ contains precisely the norm-1 linear combinations of values in $[\![A]\!]$:*

$$[\![\sharp A]\!] = ([\![A]\!]^{\perp})^{\perp} = \mathsf{Span}([\![A]\!]) \cap \mathcal{S}_1.$$

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 $\vec{v} \notin \mathsf{Span}[\![A]\!] \cap \mathcal{S}_1$. □

Theorem (Restatement of 4)**.** *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: $[\![\sharp(\sharp A)]\!] = A^{\perp^4} = A^{\perp^2} = [\![\sharp A]\!]$ □

Theorem (Restatement of 5)**.** *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 $\|(\vec{v}, \vec{w})\| = 1$. □ □

Lemma (Restatement of 10)**.** *Let X and Y be orthonormal bases of the same finite dimension, and let $\lambda x_X \,.\, \vec{t}$ be a closed λ-abstraction. Then $\lambda x_X \,.\, \vec{t} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$ if and only if for all $\vec{v_i}, \vec{v_j} \in [\![B_X]\!]$, there exist value distributions $\vec{w_i}, \vec{w_j} \in [\![\sharp B_Y]\!]$ such that,*

$$\vec{t}[\vec{v_i}/x] \twoheadrightarrow \vec{w_i} \quad \textit{and} \quad \vec{t}[\vec{v_j}/x] \twoheadrightarrow \vec{w_j}, \quad \textit{with } \vec{w_i} \perp \vec{w_j} \textit{ whenever } i \neq j.$$

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:

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

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:

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

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

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

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

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

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

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

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

Then $\sum_{i=1}^{n} \alpha_i \overrightarrow{w_i} \in [\![\sharp(\sharp B_Y)]\!] = [\![\sharp B_Y]\!]$ by 4. Since for every $\overrightarrow{u} \in [\![\sharp A]\!]$, $(\lambda x_A \,.\, \overrightarrow{t})\overrightarrow{u} \Vdash \sharp B$, we can conclude that $\lambda x_A \,.\, \overrightarrow{t} \in [\![\sharp A \Rightarrow \sharp B]\!]$. □

Before proving the soundness of the typing rules (7), we need the following results.

Theorem 9. *For all value distributions $\overrightarrow{v_1}, \overrightarrow{v_2}, \overrightarrow{w_1}, \overrightarrow{w_2}$ we have:*

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

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

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

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

□

Lemma 20. *Given a type A, two vectors $\overrightarrow{u_1}, \overrightarrow{u_2} \in [\![\sharp A]\!]$ and a scalar $\alpha \in \mathbb{C}$, there exists a vector $\overrightarrow{u_0} \in [\![\sharp A]\!]$ and a scalar $\lambda \in \mathbb{C}$ such that:*

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

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

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

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

Using the triangular inequality we observe that:

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

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

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

□

Theorem 10 (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 21. *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 $\overrightarrow{u_1}, \overrightarrow{u_2} \in [\![\sharp A]\!]$, there are value distributions $\overrightarrow{w_1}, \overrightarrow{w_2} \in [\![C]\!]$ such that:*

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

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

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 $\overrightarrow{w_1}, \overrightarrow{w_2} \in [\![C]\!]$ such that $\vec{s}\langle\sigma\rangle\langle\overrightarrow{u_1}/x\rangle_A \twoheadrightarrow \overrightarrow{w_1}$ and $\vec{s}\langle\sigma\rangle\langle\overrightarrow{u_2}/x\rangle_A \twoheadrightarrow \overrightarrow{w_2}$. In particular, we have that $\|\overrightarrow{w_1}\| = \|\overrightarrow{w_2}\| = 1$. Applying 20 four times, we know there are vectors $\overrightarrow{u_{01}}, \overrightarrow{u_{02}}, \overrightarrow{u_{03}}, \overrightarrow{u_{04}} \in [\![\sharp A]\!]$ and scalars $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ such that:

$$\overrightarrow{u_1} + \overrightarrow{u_2} = \lambda_1\overrightarrow{u_{01}}\overrightarrow{u_1} + i\overrightarrow{u_2} = \lambda_3\overrightarrow{u_{03}}$$

$$\overrightarrow{u_1} + (-1)\overrightarrow{u_2} = \lambda_2\overrightarrow{u_{02}}\overrightarrow{u_1} + (-i)\overrightarrow{u_2} = \lambda_4\overrightarrow{u_{04}}$$

From the validity of the judgement $\Delta, x_A : \sharp A \vdash \overrightarrow{s} : C$, we also know that there are value distributions $\overrightarrow{w_{01}}, \overrightarrow{w_{02}}, \overrightarrow{w_{03}}, \overrightarrow{w_{04}} \in [\![C]\!]$ such that $\overrightarrow{s}\langle\sigma\rangle\langle\overrightarrow{u_{0j}}\rangle \twoheadrightarrow \overrightarrow{w_{oj}}$ 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:

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

Using the polarization identity (10), we conclude that:

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

□

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

$$\overrightarrow{s}\langle\sigma\rangle\langle\overrightarrow{u_1}/x\rangle_{B_1}\langle\overrightarrow{v_1}/y\rangle_{B_2} \twoheadrightarrow \overrightarrow{w_1}$$
$$\overrightarrow{s}\langle\sigma\rangle\langle\overrightarrow{u_2}/x\rangle_{B_1}\langle\overrightarrow{v_2}/y\rangle_{B_2} \twoheadrightarrow \overrightarrow{w_2}$$

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

Proof. From 20 we know that there are $\overrightarrow{u_0} \in [\![\sharp A]\!], \overrightarrow{v_0} \in [\![\sharp B]\!]$ and $\lambda, \mu \in \mathbb{C}$ such that:

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

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

1. $\overrightarrow{u_1} + \lambda\overrightarrow{u_0} = \overrightarrow{u_1} + \overrightarrow{u_2} + (-1)\overrightarrow{u_1} = \overrightarrow{u_2} + 0\overrightarrow{u_1}$, so that from linearity of substitution, linearity of evaluation and uniqueness of normal forms, we get:
$$\overrightarrow{w_{1k}} + \lambda\overrightarrow{w_{0k}} = \overrightarrow{w_{2k}} + 0\overrightarrow{w_{1k}}$$
$$\overrightarrow{w_{2k}} + (-\lambda)\overrightarrow{w_{0k}} = \overrightarrow{w_{1k}} + 0\overrightarrow{w_{2k}} \qquad \text{(for all } k \in \{0, 1, 2\}\text{)}$$
2. $\overrightarrow{v_1} + \mu\overrightarrow{v_0} = \overrightarrow{v_1} + \overrightarrow{v_2} + (-1)\overrightarrow{v_1} = \overrightarrow{v_2} + 0\overrightarrow{v_1}$, so that from linearity of substitution, linearity of evaluation and uniqueness of normal forms, we get:
$$\overrightarrow{w_{j1}} + \mu\overrightarrow{w_{j0}} = \overrightarrow{w_{j2}} + 0\overrightarrow{w_{j1}}$$
$$\overrightarrow{w_{j2}} + (-\mu)\overrightarrow{w_{j0}} = \overrightarrow{w_{j1}} + 0\overrightarrow{w_{j2}} \qquad \text{(for all } j \in \{0, 1, 2\}\text{)}$$
3. $\langle\overrightarrow{u_1} \mid \overrightarrow{u_2}\rangle = 0$, so that from 21 we get $\langle\overrightarrow{w_{1k}} \mid \overrightarrow{w_{2k}}\rangle = 0$ (for all $k \in \{0, 1, 2\}$).
4. $\langle\overrightarrow{v_1} \mid \overrightarrow{v_2}\rangle = 0$, so that from 21 we get $\langle\overrightarrow{w_{j1}} \mid \overrightarrow{w_{j2}}\rangle = 0$ (for all $j \in \{0, 1, 2\}$).

From the above, we get:

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

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

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

So that we have:

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

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

Lemma 23. *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 $\overrightarrow{u_1}, \overrightarrow{u_2} \in [\![\sharp A]\!]$ and $\overrightarrow{v_1}, \overrightarrow{v_2} \in [\![\sharp B]\!]$, there are value distributions $\overrightarrow{w_1}, \overrightarrow{w_2} \in [\![C]\!]$ such that:*

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

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

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

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

And similarly that, $\langle \overrightarrow{v_1} \mid \overrightarrow{v_2} + (-\beta)\overrightarrow{v_1} \rangle = 0$. From 20, we know that there are $\overrightarrow{u_0} \in [\![\sharp A]\!]$, $\overrightarrow{v_0} \in [\![\sharp B]\!]$ and $\lambda, \mu \in \mathbb{C}$ such that:

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

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

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

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

1. $\lambda\overrightarrow{u_0} + \alpha\overrightarrow{u_1} = \overrightarrow{u_2} + (-\alpha)\overrightarrow{u_1} + \alpha\overrightarrow{u_1} = \overrightarrow{u_2} + 0\overrightarrow{u_1}$, so that from the linearity of the substitution, linearity of evaluation and uniqueness of normal forms, we get:
$$\lambda\overrightarrow{w_{0k}} + \alpha\overrightarrow{w_{1k}} = \overrightarrow{w_{2k}} + 0\overrightarrow{w_{1k}} \qquad \text{(for all } k \in \{0, 1, 2\}\text{)}$$
2. $\mu\overrightarrow{v_0} + \beta\overrightarrow{v_1} = \overrightarrow{v_2} + (-\beta)\overrightarrow{v_1} + \beta\overrightarrow{v_1} = \overrightarrow{v_2} + 0\overrightarrow{v_1}$, so that from the linearity of the substitution, linearity of evaluation and uniqueness of normal forms, we get:
$$\mu\overrightarrow{w_{j0}} + \beta\overrightarrow{w_{j1}} = \overrightarrow{w_{j2}} + 0\overrightarrow{w_{j1}} \qquad \text{(for all } j \in \{0, 1, 2\}\text{)}$$

3. $\langle \overrightarrow{u_1} \mid \lambda \overrightarrow{u_0} \rangle = \langle \overrightarrow{u_1} \mid \overrightarrow{u_2} + (-\alpha)\overrightarrow{u_1} \rangle = 0$, so that from 21 we get:
$$\langle \overrightarrow{w_{1k}} \mid \lambda \overrightarrow{w_{0k}} \rangle = 0 \qquad (\text{for all } k \in \{0,1,2\})$$
4. $\langle \overrightarrow{v_1} \mid \mu \overrightarrow{v_0} \rangle = \langle \overrightarrow{v_1} \mid \overrightarrow{v_2} + (-\beta) \rangle \overrightarrow{v_1} = 0$, so that from 21 we get:
$$\langle \overrightarrow{w_{j1}} \mid \mu \overrightarrow{w_{j0}} \rangle = 0$$
5. $\langle \overrightarrow{u_1} \mid \lambda \overrightarrow{u_0} \rangle = \langle \overrightarrow{v_1} \mid \mu \overrightarrow{v_0} \rangle = 0$ so that from 22 we get:
$$\langle \overrightarrow{w_{11}} \mid \lambda\mu \overrightarrow{w_{00}} \rangle = 0$$
(Again the equality $\langle \overrightarrow{w_{11}} \mid \lambda\mu \overrightarrow{w_{00}} \rangle$ is trivial when $\lambda = 0$ or $\mu = 0$. When $\lambda, \mu \neq 0$ we deduce from the above that $\langle \overrightarrow{u_1} \mid \overrightarrow{u_0} \rangle = \langle \overrightarrow{v_1} \mid \overrightarrow{v_0} \rangle = 0$, from which we get $\langle \overrightarrow{w_{11}} \mid \overrightarrow{w_{00}} \rangle = 0$ by 22)

From above, we get:

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

Therefore:

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

From **??** with $j = 1$ and concluding with the definition of α and β. □

Now, we can restate and prove 7.

Theorem (Restatement of 7)**.** *All the typing rules in 3.5 are valid.*

Proof. For each typing rule in 3.5 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, $x\langle\sigma\rangle = x\langle \vec{v} \rangle_B = \vec{v} \Vdash A$.

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

$$\begin{aligned}
(\sum_{i=1}^n \alpha_i (\lambda x_X \,.\, \vec{t_i}))\langle\sigma\rangle\, \vec{v} &= (\sum_{j=1}^m \beta_j (\sum_{i=1}^n \alpha_i (\lambda x_X \,.\, \vec{t_i})[\sigma_i]))\, \vec{v} \\
&= (\sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j (\lambda x_X \,.\, \vec{t_i}[\sigma_j]))\, \vec{v} \\
&\longrightarrow \sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j \vec{t_i}[\sigma_j]\langle \vec{v}/x \rangle_X
\end{aligned}$$

$$= \sum_{i=1}^{n} \alpha_i \overrightarrow{t_i}\langle\sigma\rangle\langle\overrightarrow{v}/x\rangle_X$$
$$= (\sum_{i=1}^{n} \alpha_i \overrightarrow{t_i})\langle\sigma\rangle\langle\overrightarrow{v}/x\rangle_X \qquad \text{By 8}$$

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

App If the hypotheses are valid, then:

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

From this, we can conclude that $\mathrm{dom}^\sharp(\Gamma, \Delta) \subseteq \mathrm{FV}(\overrightarrow{s}\,\overrightarrow{t}) \subseteq \mathrm{dom}(\Gamma, \Delta)$. 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:

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

Then we can conclude that $(\overrightarrow{t}\,\overrightarrow{s})\langle\sigma\rangle \Vdash B$.

Pair If the hypotheses are valid, then:

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

From this, we can conclude that $\mathrm{dom}^\sharp(\Gamma, \Delta) \subseteq \mathrm{FV}((\overrightarrow{s}, \overrightarrow{t})) \subseteq \mathrm{dom}(\Gamma, \Delta)$. 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:

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

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

LetPair If the hypotheses are valid, then:

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$ and $\vec{t}\langle\sigma_\Gamma\rangle \Vdash A \times B \ \forall \sigma_\Gamma \in [\![\Gamma]\!]$
- $\mathrm{dom}^\sharp(\Delta, x^X : A, y^Y : B) \subseteq \mathrm{FV}(\vec{s})$
- $\mathrm{FV}(\vec{s}) \subseteq \mathrm{dom}(\Delta, x^X : A, y^Y : B)$
- $\vec{s}\langle\sigma_\Delta\rangle \Vdash C \ \forall \sigma_\Delta \in [\![\Delta, x^X : A, y^Y : B]\!]$

From this, we can conclude that:

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

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:

$$
\begin{aligned}
&(\mathsf{let}_{(X,Y)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma\rangle = \\
&(\mathsf{let}_{(X,Y)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&= (\sum_{i=1}^{n} \alpha_i (\mathsf{let}_{(X,Y)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s})[\sigma_{\Gamma i}])\langle\sigma_{\Delta j}\rangle \\
&\equiv (\mathsf{let}_{(X,Y)}\ (x, y) = \sum_{i=1}^{n} \alpha_i [\sigma_{\Gamma i}]\vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma_\Delta\rangle \\
&= (\mathsf{let}_{(X,Y)}\ (x, y) = \vec{t}\langle\sigma_\Gamma\rangle\ \mathsf{in}\ \vec{s})\langle\sigma_\Delta\rangle \\
&\twoheadrightarrow (\mathsf{let}_{(X,Y)}\ (x, y) = e^{i\theta}(\vec{v}, \vec{w})\ \mathsf{in}\ \vec{s})\langle\sigma_\Delta\rangle \\
&\qquad\qquad \text{Where: } \vec{v} \in [\![A]\!], \vec{w} \in [\![B]\!] \\
&\longrightarrow e^{i\theta_1}(\vec{s}\langle\sigma_\Delta\rangle\langle(\vec{v}, \vec{w})/x \otimes y\rangle_{X\otimes Y}) \\
&= e^{i\theta_1}(\vec{s}\langle\sigma_\Delta\rangle\langle\vec{v}/x\rangle_X\langle\vec{w}/y\rangle_Y)
\end{aligned}
$$

$$\twoheadrightarrow e^{i\theta_1}(e^{i\theta_2}\,\vec{u}) \qquad \text{where: } \vec{u} \in [\![C]\!]$$
$$\equiv e^{i\theta}\,\vec{u} \qquad \text{where: } \theta = \theta_1 + \theta_2$$

Since $\langle\sigma_\Delta\rangle\langle\vec{v}/x\rangle_X\langle\vec{w}/y\rangle_Y \in [\![\Delta, x^X : A, y^Y : B]\!]$, then we can conclude that $(\mathsf{let}_{(X,Y)}\ (x,y) = \vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma\rangle \Vdash C$.

LetTens If the hypotheses are valid then:

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$ and $\vec{t}\langle\sigma\rangle \Vdash \sharp(A \times B)\ \forall\sigma \in [\![\Gamma]\!]$
- $\mathrm{dom}^\sharp(\Delta, x^X : \sharp A, y^Y : \sharp B) \subseteq \mathrm{FV}(\vec{s})$
- $\subseteq \mathrm{dom}(\Delta, x^X : \sharp A, y^Y : \sharp B)$
- $\vec{s}\langle\sigma\rangle \Vdash \sharp C\ \forall\sigma \in [\![\Delta, x^X : \sharp A, y^Y : \sharp B]\!]$

From this we can conclude that:

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

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, $\vec{t}\langle\sigma_\Gamma\rangle \Vdash \sharp(A \times B)$, from 3 we have that:

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

With:

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

Then:

$$\begin{aligned}
&(\mathsf{let}_{(X,Y)}\ (x,y) = \vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma\rangle \\
&= \mathsf{let}_{(X,Y)}\ (x,y) = \vec{t}\ \mathsf{in}\ \vec{s}\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&= (\sum_{i=1}^{n} \alpha_i \mathsf{let}_{(X,Y)}\ (x,y) = \vec{t}\ \mathsf{in}\ \vec{s}\ [\sigma_{\Gamma i}])\langle\sigma_\Delta\rangle \\
&\equiv (\mathsf{let}_{(X,Y)}\ (x,y) = \sum_{i=1}^{n} \alpha_i\, \vec{t}\ [\sigma_{\Gamma i}]\ \mathsf{in}\ \vec{s})\langle\sigma_\Delta\rangle \\
&= (\mathsf{let}_{(X,Y)}\ (x,y) = \vec{t}\langle\sigma_\Gamma\rangle\ \mathsf{in}\ \vec{s})\langle\sigma_\Delta\rangle \\
&\twoheadrightarrow (\mathsf{let}_{(X,Y)}\ (x,y) = e^{i\theta_1}\,\vec{u}\ \mathsf{in}\ \vec{s})\langle\sigma_\Delta\rangle \\
&\longrightarrow e^{i\theta_1}(\vec{s}\langle\sigma_\Delta\rangle\langle\vec{u}/x \otimes y\rangle_{X\otimes Y}) \\
&= e^{i\theta_1}(\sum_{k=1}^{l} \gamma_k\, \vec{s}\langle\sigma_\Delta\rangle\langle\vec{v_k}/x\rangle_X\langle\vec{u_k}/y\rangle_Y) \\
&\twoheadrightarrow e^{i\theta_1}(\sum_{k=1}^{l} \gamma_k e^{i\rho_k}\vec{w_k}) \quad \text{where: } \vec{w_k} \in [\![C]\!]
\end{aligned}$$

It remains to be seen that the term has norm-1, $\| \sum_{k=1}^{l} \gamma_k e^{i\rho_k} \overrightarrow{w_k} \| = 1$. For that, we observe:

$$
\begin{aligned}
&\| \sum_{k=1}^{l} \gamma_k e^{i\rho_k} \overrightarrow{w_k} \| \\
&= \langle \sum_{k=1}^{l} \alpha_i e^{i\rho_k} \overrightarrow{w_k} \mid \sum_{k'=1}^{l} \gamma_{k'} e^{i\rho_{k'}} \overrightarrow{w_{k'}} \rangle \\
&= \sum_{k=1}^{l} \sum_{k'}^{l} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle \overrightarrow{w_k} \mid \overrightarrow{w_{k'}} \rangle \\
&= \sum_{k=1}^{l} \sum_{k'=1}^{l} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle \overrightarrow{v_k} \mid \overrightarrow{v_{k'}} \rangle \langle \overrightarrow{u_k} \mid \overrightarrow{u_{k'}} \rangle \quad \text{(from 23)} \\
&= \sum_{k=1}^{k} \sum_{k'=1}^{l} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle (\overrightarrow{u_k}, \overrightarrow{v_k}) \mid (\overrightarrow{u_{k'}}, \overrightarrow{v_{k'}}) \rangle \quad \text{(from 9)} \\
&= \sum_{k=1}^{n} \overline{\gamma_k e^{i\rho_k}}\ \gamma_k e^{i\rho_k} \langle (\overrightarrow{v_k}, \overrightarrow{u_k}) \mid (\overrightarrow{v_k}, \overrightarrow{u_k}) \rangle \\
&\quad + \sum_{k,k'=1;k\neq k'}^{n} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle (\overrightarrow{v_k}, \overrightarrow{u_k}) \mid (\overrightarrow{v_{k'}}, \overrightarrow{u_{k'}}) \rangle \\
&= \sum_{k=1}^{n} \overline{\gamma_k e^{i\rho_k}}\ \gamma_k e^{i\rho_k} + 0 \\
&= \sum_{k=1}^{l} |\gamma_k|^2 |e^{i\rho_k}|^2 = 1
\end{aligned}
$$

Then $\sum_{k=1}^{l} \gamma_k e^{i\rho_k} \overrightarrow{w_k} \in [\![\sharp C]\!]$. Finally, we can conclude that: $(\mathsf{let}_{(X,Y)}\ (x,y) = \overrightarrow{t}\ \mathsf{in}\ \overrightarrow{s})\langle\sigma\rangle \Vdash \sharp C$

Case If the hypotheses are valid then:

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

From this we can conclude that:

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

Then, 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, $\overrightarrow{t}\langle\sigma_\Gamma\rangle \twoheadrightarrow e^{i\theta_1} \overrightarrow{v_k}$ for some $k \in \{1, \cdots, n\}$. From the second hypothesis we have that $\overrightarrow{s_i}\langle\sigma_\Delta\rangle \twoheadrightarrow e^{i\rho_i} \overrightarrow{u_i} \in [\![A]\!]$ for $i \in \{1, \cdots, n\}$. Therefore:

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

Since we pose no restriction on k, we can conclude that: $(\mathsf{case}\ \overrightarrow{t}\ \mathsf{of}\{\overrightarrow{v_1} \mapsto \overrightarrow{s_1} \mid \cdots \mid \overrightarrow{v_n} \mapsto \overrightarrow{s_n}\})\langle\sigma\rangle \Vdash A$

UnitCase If the hypotheses are valid, then:

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

From this we can conclude that:

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

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

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

$$
\begin{aligned}
&= e^{i\theta_1}(\sum_{j=1}^{n}\delta_j(\sum_{i=1}^{n}\beta_i\vec{s_i})[\sigma_{\Delta j}]) \\
&\equiv e^{i\theta_1}(\sum_{i,j=1}^{n}\beta_i\delta_j\vec{s_i}[\sigma_{\Delta j}]) \\
&= e^{i\theta_1}(\sum_{i=1}^{n}\beta_i\vec{s_i}\langle\sigma_\Delta\rangle) \\
&\twoheadrightarrow e^{i\theta_1}(\sum_{i=1}^{n}\beta_i e^{i\rho_i}\vec{u_i})
\end{aligned}
$$

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

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

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, $\vec{t_i}\langle\sigma\rangle \twoheadrightarrow e^{i\rho_i}\vec{v_i}$ 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:

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

$$\twoheadrightarrow \sum_{i=1}^{n} \alpha_i e^{i\rho_i} \vec{v_i}$$

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

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

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

Contr If the hypothesis is valid, we have that $\mathrm{dom}^\sharp(\Gamma, x^X : B_X, y^X : B_X) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma, x^X : B_X, y^X : B_X)$ and given $\sigma \in [\![\Gamma, x^X : B_X, y^X : B_X]\!]$, then $\vec{t}\langle\sigma\rangle \in [\![B]\!]$. Then, we have that $\mathrm{dom}^\sharp(\Gamma, x^X : B_X, y^X : B_X) = \mathrm{dom}^\sharp(\Gamma, x^X : B_X)$. Therefore:

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

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

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

Since $\langle \vec{v}/y \rangle_X \langle \vec{v}/x \rangle_X \langle\sigma\rangle \in [\![\Gamma, x^X : B_X, y^X : B_X]\!]$, we get: $\vec{t}\langle \vec{v}/y \rangle_X \langle \vec{v}/x \rangle_X \langle\sigma_\Gamma\rangle \twoheadrightarrow e^{i\theta}\vec{w} \in [\![B]\!]$ Then we can finally conclude that $\vec{t}[x/y]\langle\sigma\rangle \Vdash B$.

Weak Given $\sigma \in [\![\Gamma, x^X : B_X]\!]$, we observe that $\langle\sigma\rangle = \langle\sigma_\Gamma\rangle \langle \vec{v}/x \rangle_X$ for some $\sigma_\Gamma \in [\![\Gamma]\!]$ and $\vec{v} \in [\![B_X]\!]$. 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:

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

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

Sub Trivial since the set of realizers of A is included in the set of realizers of B.

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

Phase It follows from the definition of type realizers.

□

Lemma 24 (Substitution)**.** *Let Γ, Δ be contexts, A and B types, and X an orthonormal basis. If $\Gamma, x^X : A \vdash \overrightarrow{t} : B$ and $\Delta \vdash \overrightarrow{v} : A$ can be derived using the set of rules in 3.5, and the substitution $\overrightarrow{t}\langle\overrightarrow{v}/x\rangle_X$ is defined, then $\Gamma, \Delta \vdash \overrightarrow{t}\langle\overrightarrow{v}/x\rangle_X : B$ can also be derived by the same set of rules.*

Proof. By induction on $\overrightarrow{t}$.

$x \notin \mathrm{FV}(\overrightarrow{t})$**:** Then $x \notin \mathrm{dom}^\sharp(\Gamma, x^X : A)$. By a straightforward generation lemma we have that $\Gamma \vdash \overrightarrow{t} : B$, and we can derive $\Gamma, \Delta \vdash \overrightarrow{t} : B$ via the Weak rule. Notice that if $x \notin \mathrm{FV}(\overrightarrow{t})$, every variable in Δ has a non-linear type and $\overrightarrow{t}\langle\overrightarrow{v}/x\rangle_X = \overrightarrow{t}$.

$\overrightarrow{t} = x$**:** Then every variable in Γ has a non-linear type and $A \leq B$. This means we can derive $\Gamma, \Delta \vdash \overrightarrow{v} : B$, by rules Weak and Sub.

$\overrightarrow{t} = (\lambda y_Y \,.\, \overrightarrow{s})$**:** Then $\Gamma, y^Y : C, x^X : A \vdash \overrightarrow{s} : D$, with $C \Rightarrow D \leq B$. By induction hypothesis, $\Gamma, y^Y : C \vdash \overrightarrow{s}\langle\overrightarrow{v}/x\rangle_X : D$. This means we can derive $\Gamma, \Delta \vdash (\lambda y_Y \,.\, \overrightarrow{s})\langle\overrightarrow{v}/X\rangle_X : B$, by rules Sub and UnitLam.

$\overrightarrow{t} = \overrightarrow{s_1}\,\overrightarrow{s_2}$**:** Then we have that $\Gamma_1 \vdash \overrightarrow{s_1} : C \Rightarrow D$, and $\Gamma_2 \vdash \overrightarrow{s_2} : C$ with $D \leq B$ and $\Gamma_1, \Gamma_2 = \Gamma, x^X : A$. We consider the case where $\Gamma_1 = \Gamma'_1, x^X : A$. Then, by inductive hypothesis $\Gamma'_1 \Delta \vdash \overrightarrow{s_1}\langle\overrightarrow{v}/x\rangle_X : C \Rightarrow D$. This means we can derive $\Gamma'_1, \Gamma_2, \Delta \vdash (\overrightarrow{s_1}\,\overrightarrow{s_2})\langle\overrightarrow{v}/x\rangle_X : B$, by rules Sub and App. The case where $\Gamma_2 = \Gamma'_2, x^X : A$ is analogous.

$\overrightarrow{t} = (s_1, s_2)$**:** Then we have that $\Gamma_1 \vdash s_1 : C$, and $\Gamma_2 \vdash s_2 : D$ with $C \times D \leq B$ and $\Gamma_1, \Gamma_2 = \Gamma, x^X : A$. We consider the case where $\Gamma_1 = \Gamma'_1, x^X : A$. Then, by inductive hypothesis $\Gamma'_1, \Delta \vdash s_1 : C$. This means we can derive $\Gamma'_1, \Gamma_2, \Delta \vdash (s_1, s_2)\langle\overrightarrow{v}/x\rangle_X : B$, by rules Sub and Pair. The case where $\Gamma_2 = \Gamma'_2, x^X : A$ is analogous.

$\overrightarrow{t} = \mathsf{let}_{(Y,Z)}\ (y, z) = \overrightarrow{s_1}\ \mathsf{in}\ \overrightarrow{s_2}$**:** Then we have two possibilities:

1. Either, $\Gamma_1 \vdash \overrightarrow{s_1} : C \times D$, and $\Gamma_2, y^Y : C, z^Z : D \vdash \overrightarrow{s_2} : E$ with $E \leq B$.
2. Or, $\Gamma_1 \vdash \overrightarrow{s_1} : \sharp(C \times D)$, and $\Gamma_2, y^Y : \sharp C, z^Z : \sharp D \vdash \overrightarrow{s_2} : E$ with $\sharp E \leq B$.

In either way, we consider the case where $\Gamma_1 = \Gamma'_1, x^X : A$. Then, by inductive hypothesis $\Gamma'_1\,\Delta \vdash \overrightarrow{s_1}\langle\overrightarrow{v}/x\rangle : C \times D$ in case 1 ($\sharp(C \times D)$ in case 2). This means we can derive $\Gamma'_1, \Gamma_2, \Delta \vdash (\mathsf{let}_{(Y,Z)}\ (y, z) = \overrightarrow{s_1}\ \mathsf{in}\ \overrightarrow{s_2})\langle\overrightarrow{v}/x\rangle_X : B$ by rules Sub and LetPair (or, LetTens). The case where $\Gamma_2 = \Gamma'_2, x^X : A$ is analogous.

$\overrightarrow{t} = \mathsf{case}\ \overrightarrow{s}\ \mathsf{of}\{\overrightarrow{w_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{w_n} \mapsto \overrightarrow{r_n}\}$**:** Then we have two possibilities:

1. Either, $\Gamma_1 \vdash \overrightarrow{s_1} : B_{\{\overrightarrow{v_i}\}_{i=1}^n}$, and for all $i \in \{0, \cdots, n\}$, $\Gamma_2 \vdash \overrightarrow{s_i} : C$ with $C \leq B$.
2. Or, $\Gamma_1 \vdash \overrightarrow{s_1} : \sharp B_{\{\overrightarrow{v_i}\}_{i=1}^n}$, and for all $i, j \in \{0, \cdots, n\}$, with $i \neq j$ $\Gamma_2 \vdash \overrightarrow{s_i} \perp \overrightarrow{s_j} : C$ with $\sharp C \leq B$.

In either way, we consider the case where $\Gamma_1 = \Gamma'_1, x^X : A$. Then, by inductive hypothesis $\Gamma'_1\,\Delta \vdash \overrightarrow{s_1}\langle\overrightarrow{v}/x\rangle : B_{\{\overrightarrow{v_i}\}_{i=1}^n}$ in case 1 ($\sharp B_{\{\overrightarrow{v_i}\}_{i=1}^n}$ in case 2). This means we can derive $\Gamma'_1, \Gamma_2, \Delta \vdash (\mathsf{case}\ \overrightarrow{s}\ \mathsf{of}\{\overrightarrow{w_1} \mapsto \overrightarrow{r_1} \mid \cdots \mid \overrightarrow{w_n} \mapsto \overrightarrow{r_n}\})\langle\overrightarrow{v}/x\rangle_X : B$ by rules Sub and Case (or, UnitCase). Since orthogonality is preserved by substitutions in $[\![\Gamma_2]\!]$, the case where $\Gamma_2 = \Gamma'_2, x^X : A$ is analogous.

$\overrightarrow{t} = \sum_{i=1}^n \alpha_i \overrightarrow{s_i}$**:** Then we have two posibilities:

1. For all $i \in \{0, \cdots, n\}, \vec{s_i} = (\lambda y_Y \,.\, \vec{r_i})$, and:
   $\Gamma,\, x^X : A \vdash \sum_{i=1}^n \alpha_i (\lambda y_Y \,.\, \vec{r_i}) : C \Rightarrow D$ with $C \Rightarrow D \leq B$.
2. For all $i \in \{0, \cdots, n\}$, with $i \neq j$, $\Gamma,\, x^X : A \vdash \vec{s_i} \perp \vec{s_j} : C$, $\sum_{i=1}^n |\alpha_i|^2 = 1$, and $\sharp C \leq B$.

In case 1, by inductive hypothesis we have that:
$\Gamma,\, \Delta,\, y^Y : C \vdash \sum_{i=1}^n \alpha_i \vec{r_i} \langle \vec{v}/x \rangle_X : D$. This means we can derive
$\Gamma, \Delta \vdash \sum_{i=1}^n \alpha_i (\lambda y_Y \,.\, \vec{r_i}) \langle \vec{v}/x \rangle_X : B$ by rules Sub and UnitLam.

In case 2, by inductive hypothesis we have that for all $i \in \{0, \cdots, n\}$, and $i \neq j$ $\Gamma, \Delta \vdash \vec{s_i} \langle \vec{v}/x \rangle_X \perp \vec{s_j} \langle \vec{v}/x \rangle_X : C$. Since orthogonality is preserved by substitutions in $[\![\Gamma, \Delta]\!]$, this means we can derive $\Gamma, \Delta \vdash \sum_{i=1}^n \alpha_i \vec{s_i} \langle \vec{v}/x \rangle_X : B$ by rules Sub and Sum.

$\vec{t} = \alpha \vec{s}$**:** Then $\alpha = e^{i\theta}$, and $\Gamma,\, x^X : A \vdash \vec{s} : C$ with $C \leq B$. By induction hypothesis, $\Gamma \vdash \vec{s} \langle \vec{v}/x \rangle_X : C$. This means we can derive $\Gamma, \Delta \vdash e^{i\theta} \vec{s} \langle \vec{v}/X \rangle_X :$ B, by rules Sub and Phase. □

Theorem (Restatement of 8)**.** *If $\Gamma \vdash \vec{t} : A$ can be derived using the set of rules in 3.5 and $\vec{t} \to \vec{u}$, then $\Gamma \vdash \vec{u} : A$ can also be derived by the same set of rules.*

Proof. We proceed by induction on the derivation of the elementary reduction $\rightsquigarrow$. The congruence closure to obtain $\longrightarrow$ is handled trivially via the Equiv rule. We only give the basis cases as the inductive cases (the contextual cases) are straightforward.

- Let $(\lambda x_X \,.\, \vec{t})\, \vec{v} \rightsquigarrow \vec{t} \langle \vec{v}/x \rangle_X$: Assume $\Gamma, \Delta \vdash (\lambda x_X \,.\, \vec{t})\, \vec{v} : B$. By App, we have $\Gamma \vdash \lambda x_X \,.\, \vec{t} : A \Rightarrow B$ and $\Delta \vdash \vec{v} : A$. From $\Gamma \vdash \lambda x_X \,.\, \vec{t} : A \Rightarrow B$ and UnitLam, we get $\Gamma, x^X : A \vdash \vec{t} : B$. By 24, using $\Delta \vdash \vec{v} : A$ and the fact that $\vec{t} \langle \vec{v}/x \rangle_X$ is defined, we conclude $\Gamma, \Delta \vdash \vec{t} \langle \vec{v}/x \rangle_X : B$, as required.
- Let $\mathsf{let}_{(X,Y)}\ (x, y) = \vec{v}\ \mathsf{in}\ \vec{t} \rightsquigarrow \vec{t} \langle \vec{v}/x \otimes y \rangle_{X \otimes Y}$: Assume $\Gamma, \Delta \vdash \mathsf{let}_{(X,Y)}\ (x, y) = \vec{v}\ \mathsf{in}\ \vec{t} : C$. By LetPair we have $\Gamma \vdash \vec{v} : A \times B$ and $\Delta,\, x^X : A,\, y^Y : B \vdash \vec{t} : C$. Decompose $\vec{v}$ (in $X \otimes Y$) as required by the definition of $\langle \vec{v}/x \otimes y \rangle_{X \otimes Y}$; by 24 applied twice (first for x, then for y), we conclude $\Gamma, \Delta \vdash \vec{t} \langle \vec{v}/x \otimes y \rangle_{X \otimes Y} : C$.
- Let $\mathsf{case}\ \vec{v_k}\ \mathsf{of} \{\vec{v_1} \mapsto \vec{t_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{t_n}\} \rightsquigarrow \vec{t_k}$: Assume $\Gamma, \Delta \vdash \mathsf{case}\ \vec{v_k}\ \mathsf{of} \{\vec{v_1} \mapsto \vec{t_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{t_n}\} : A$. By Case we have $\Gamma \vdash \vec{v_k} : B_{\{\vec{v}_i\}_{i=1}^n}$ and $\Delta \vdash \vec{t}_i : A$ for all i. Thus, we are done.
- Let $\mathsf{case} \sum_{i=1}^n \alpha_i \vec{v_i}\ \mathsf{of} \{\vec{v_1} \mapsto \vec{t_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{t_n}\} \rightsquigarrow \sum_{i=1}^n \alpha_i \vec{t_i}$: Assume $\Gamma, \Delta \vdash \mathsf{case} \sum_{i=1}^n \alpha_i \vec{v_i}$ $\vec{t_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{t_n}\} : A$. By UnitCase we have $\Gamma \vdash \vec{t} : \sharp B_{\{\vec{v}_i\}}$ and, for all $i \neq j$, $\Delta \vdash \vec{t}_i \perp \vec{t}_j : A$. Then the reduct $\sum_{i=1}^n \alpha_i \vec{t_i}$ is typed by Sum as $\sharp A$ (using the orthogonality premises and the normalisation condition ensured by the semantics of $\sharp$), hence $\Gamma, \Delta \vdash \sum_{i=1}^n \alpha_i \vec{t_i} : \sharp A$. □

List of Figures

List of Tables

Bibliography

[1] S. Alves, B. Dundua, M. Florido, and T. Kutsia. Pattern-based calculi with finitary matching. *Logic Journal of the IGPL*, 26:203–243, 2018. 1.3.2

[2] 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

[3] 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

[4] 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

[5] O. Bournez and F. Garnier. Proving positive almost-sure termination. In J. Giesl, editor, *Term Rewriting and Applications (RTA 2005)*, volume 3467 of *Lecture Notes in Computer Science*, pages 323–337. Springer, 2005. 1.1

[6] O. Bournez and C. Kirchner. Probabilistic Rewrite Strategies. Applications to ELAN. In S. Tison, editor, *Rewriting Techniques and Applications (RTA 2002)*, volume 2378 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2002. 1.1

[7] 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

[8] 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

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

[10] 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

[11] 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

[12] U. Dal Lago, G. Guerrieri, and W. Heijltjes. Decomposing probabilistic lambda-calculi. In J. Goubault-Larrecq and B. König, editors, *Foundations of Software Science and Computation Structures (FoSSaCS 2020)*, volume 12077 of *Lecture Notes in Computer Science*, pages 136–156. Springer, 2020. 1.3.1

[13] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, January 2020. tex.ids: debeaudrapZXCalculusLanguage2020a arXiv: 1704.08670. 2.1

[14] 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.

[15] Cirq Developers. Cirq, August 2021. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors. 2.1

[16] A. Díaz-Caro. A lambda calculus for density matrices with classical and probabilistic controls. In B.-Y. E. Chang, editor, *Programming Languages and Systems (APLAS 2017)*, volume 10695 of *Lecture Notes in Computer Science*, pages 448–467. Springer, 2017. 1.3.2, 1.5

[17] A. Díaz-Caro and G. Martínez. Confluence in probabilistic rewriting. In S. Alves and R. Wassermann, editors, *Logical and Semantic Frameworks with Applications (LSFA 2017)*, volume 338 of *Electronic Notes in Teoretical Computer Science*, pages 115–131. Elsevier, 2018. 1.1, 1.2

[18] Alejandro Díaz-Caro. Towards a computational quantum logic: An overview of an ongoing research program, 2025. Invited paper.

[19] 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, 3.4.2, 3.6

[20] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus, 2019. 2.1

[21] 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. 3.1

[22] Alejandro Díaz-Caro and Nicolas A. Monzon. A quantum-control lambda-calculus with multiple measurement bases, 2025.

[23] MD SAJID ANIS et al. Qiskit: An open-source framework for quantum computing, 2021. 2.1

[24] C. Faggian. Probabilistic Rewriting: Normalization, Termination, and Unique Normal Forms. In H. Geuvers, editor, *Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:25. Schloss Dagstuhl, 2019. 1.1

[25] 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

[26] 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

[27] 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

[28] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945. 3.1

[29] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011. 2.5

[30] Simon Perdrix. *Quantum Entanglement Analysis Based on Abstract Interpretation*, page 270–282. Springer Berlin Heidelberg, 2008. 3.1

[31] R. Romero. Una extensión polimórfica para los λ-cálculos cuánticos λ_ρ y λ_ρ°. Master's thesis, Universidad de Buenos Aires, Argentina, 2020. 1.5

[32] Damian S. Steiger, Thomas Häner, and Matthias Troyer. Projectq: an open source software framework for quantum computing. *Quantum*, 2:49, Jan 2018. 2.1

[33] John van de Wetering. Zx-calculus for the working quantum computer scientist, 2020. 2.1, 2.2.1