Categorical Aspects of Quantum Lambda Calculus

Ludwig Maximilian University of Munich
Faculty of Mathematics, Statistics and Computer Science
Mathematical Institute

Ioannis Andreou

Master's Thesis

Submitted by:
Ioannis Andreou
Matriculation number: 12343130
Study program: Mathematics M.Sc.

Supervisor:
PD. Dr. Iosif Petrakis

Declaration of Authorship

I hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text. This also applies to all graphics, images and tables included in the thesis.

Munich, 31.10.23 .....

Ioannis Andreou

Abstract

Quantum computers intrinsically differ from classical ones and consequently the theory of classical computations can not be applied to them. Therefore in this thesis a lambda calculus for quantum computations is presented. As an introduction the fundamentals of category theory and quantum computation are being explained. Afterwards we analyze the mathematics of quantum mechanics categorically and use the findings to draw connections to logic. For that matter light is also shed on linear logic and its properties. Finally a quantum lambda calculus is presented, with a typing system based on linear logic. The calculus is analyzed thoroughly and its operational semantics are clarified. Additionally a comparison of its categorical semantics to the categories found in quantum mechanics is made.

Acknowledgements

I would like to thank the following people:

- Professor Schuster and Professor Zorzi for inviting me to Verona and showing interest in my work.
- Professor Petrakis, for his patience, care and support on numerous occasions. I could not have imagined a better supervisor.
- My family for their endless love and trust.

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 The Aim of this Thesis 1
1.2 Related Work and Distinguishing Features 2
1.3 Fundamental Notions of Category Theory 2
1.4 Introduction to Quantum Computation 4
1.4.1 Qubits and States 4
1.4.2 Operators, Observables and Gates 6
1.4.3 No Cloning and No Deletion 8
1.4.4 Advantages and Limitations 9

2 The Internal Logic of Quantum Mechanics **11**
2.1 A Hilbert Space Category 11
2.1.1 Morphisms of Hilbert Spaces 11
2.1.2 Hilb as a †-Category 13
2.1.3 Monoidal Categories 14
2.1.4 Closed Categories 17
2.1.5 Compact Categories 18
2.1.6 Dagger Compact Closed Categories 19
2.2 Linear Logic 20
2.2.1 A Background in Logic and Proof Theory 21
2.2.2 A Sequent Calculus for Linear Logic 24
2.2.3 Natural Deduction for Linear Logic 25
2.2.4 Closed Symmetric Monoidal Theories 26
2.2.5 Further properties of Linear Logic 29

3 Quantum Lambda Calculus **33**
3.1 Construction of a Quantum Lambda Calculus 33
3.1.1 Terms 33
3.1.2 Types and Subtyping 37
3.1.3 Typing Rules 39
3.1.4 Operational Semantics 41
3.1.5 Equational Logic 45

3.2 Categorical Semantics 63
3.2.1 Categories of Types 63
3.2.2 Product Structure 64
3.2.3 Semantics of the Function Type 75
3.2.4 Semantics of the Bang Operator "!" 78
3.2.5 Combined Structure 80

4 Conclusions and Future Work **83**
4.1 Current and Future Research 83
4.1.1 Parallel-Non-Parallel Logic 83
4.1.2 Dependent Linear Type Theory 84
4.2 Summary and Conclusions 85

Chapter 1

Introduction

1.1 The Aim of this Thesis

The impact of the classical computer on our society can not be understated. It has not only become an irreplacable part of our everyday lifes, but also made progress in sciences and technologies significantly easier. Along with the computer a theory of computation was developed, which build the theoretical foundations of programing languages and enabled scientists to analyze the computability of many problems: the lambda calculus. With the role of the classical computer being as big as it is, it is no surprise that the developement of a new type of computer, the quantum computer, has sparked a lot of excitement. In contrast to its classical counterpart, a quantum computers fundamental unit of information, the qubit, obeys the laws of quantum mechanics, allowing for the exploitation of many quantum effects. In particular a qubit can be in a superposition of many states at once, and different qubits can interact with eachother through constructive or destructive interference. A large number of algorithms have been developed which use these effects to solve problems, which would take a classical computer an unpractically long time to solve. However the fundamental differences between classical and quantum computers also entail, that the lambda calculus classically used to describe computations can not be applied to quantum computers. In this thesis we thus analyze the newly developed Quantum Lambda Calculus. The Quantum Lambda Calculus we consider here is a modified version of the one developed by P. Selinger and B. Valiron [25, 22, 23], an outline of the similarities and modifications is found in section 1.2.

The aim of this thesis is to show why this Quantum Lambda Calculus is appropriate for describing computations in a quantum computer. For that in chapter 2 we draw connections between quantum mechancis and mathematical logic, more precisely linear logic. In chapter 3 we then present the construction of a typed lambda calculus based on the previous results. Throughout this thesis we work largely in a categorical framework, which enables us to translate between the various fields easily.

In order to make this content available to a broader audience, introduce basics of category theory and quantum computation later in this chapter.

1.2 Related Work and Distinguishing Features

Chapter 2 of this thesis follows in large parts the developements of J.C. Baez and M. Stay in their "Rosetta Stone" paper [6]. In comparison to that, it is here explicitly shown that linear logic suffices their notion of a *closed symmetric monoidal theory*.

The lambda calculus presented in chapter 3 was developed by B. Valiron in his PhD thesis [25]. In comparison to their system, the system in this thesis additionally contains a coproduct. A quantum lambda calculus with coproducts has already been defined in papers by P. Selinger and B. Valiron [23, 22], but their system in based on affine logic (linear logic with weakening) while here we use linear logic (without weakening), since that is more in line with the other parts of the thesis. Furthermore most results in their paper were only proven for the fragment of their system that does not include the coproduct. In this thesis most of those results are extended to encompass the coproduct, too. Additionally and most notably in section 3.1.5 we add to the *substitution lemma* a *reverse substitution lemma*. The substitution lemma, which can already be found in the mentioned sources, is sufficient to show that applying β- and η-reduction to the term of a judgement, does not affect the validity of that judgement. The reverse substitution lemma, that we showcase in this thesis, suffices to prove, that the corresponding expansions don't affect the validity of the judgements either. Since an equivalence relation is then build upon those reductions/expansions, showing that one may traverse them in both directions is in the author's view appropriate if not necessary.

The logic sketched in section 4.1.1 has already been developed by R. Duncan, S. Abramsky and B. Coecke [11, 3, 2]. Particularily the logical rules are found in a similar form in [10]. The rules were found independently by the author, before noticing that the system coincides with an already existing one. In comparison to their work, the author added an explanation of the logic not depending on quantum physics, but rather on availability of resources as commonly done for linear logic.

1.3 Fundamental Notions of Category Theory

Definition 1.1 (Category)**.**
A **category** $\mathcal{C}$ consists of

- A collection of **objects** A,B,C,...
- A collection of **arrows** or **morphisms** f, g, h such that:
- for each arrow f there are objects $\mathrm{dom}(f)$ and $\mathrm{cod}(f)$ called the **domain** and **codomain** of f and we write $f : A \longrightarrow B$ whenever $\mathrm{dom}(f) = A$ and $\mathrm{cod}(f) = B$.
- For each pair of arrows f, g s.t. $\mathrm{cod}(f) = \mathrm{dom}(g)$, say $f : A \longrightarrow B$ and $g : B \longrightarrow C$, there is a composite arrow $g \circ f : A \longrightarrow C$, sometimes also simply written as gf.
- For each object A, there is an **identity arrow** $id_A : A \longrightarrow A$

- Composition is associative i.e. for f, g as above and $h : C \longrightarrow D$ we have

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

- Identity arrows are left and right units of composition i.e. for all $f : A \longrightarrow B$

$$f \circ id_A = f = id_B \circ f$$

Definition 1.2 (Isomorphism)**.**
A morphism $f : A \longrightarrow B$ is called an **isomorphism** if there exists a morphism $g : B \longrightarrow A$ such that $gf = id_A$ and $fg = id_B$ and we then say that g is the **inverse** of f.

Definition 1.3 (Functor)**.**
A **functor** $F : \mathcal{C} \longrightarrow \mathcal{D}$ is a map from a category $\mathcal{C}$ to a $\mathcal{D}$ meaning it sends

- objects A in $\mathcal{C}$ to objects $F(A)$ in $\mathcal{D}$
- arrows $f : A \longrightarrow B$ in $\mathcal{C}$ to arrows $F(f) : F(A) \longrightarrow F(B)$ in $\mathcal{D}$. (Note in particular, that $dom(F(f)) = F(dom(f))$ and analogously for cod)

such that identities and composition are preserved:

- $F(id_A) = id_{F(A)}$ for any A in $\mathcal{C}$
- $F(g \circ f) = F(g) \circ F(f)$ for any $f : A \longrightarrow B$, $g : B \longrightarrow C$ in $\mathcal{C}$

Definition 1.4 (Natural transformation)**.**
Let $F, G : \mathcal{C} \longrightarrow \mathcal{D}$ be two functors, then a **natural transformation** from F to G, denoted by $\vartheta : F \Longrightarrow G$, assigns to each object A in $\mathcal{C}$ a morphism $\vartheta_A : F(A) \longrightarrow G(A)$ such that for any $f : A \longrightarrow B$ in $\mathcal{C}$ we have $\vartheta_B \circ F(f) = G(f) \circ \vartheta_A$ i.e. the following diagram commutes:

$$\begin{array}{ccc} F(A) & \xrightarrow{F(f)} & F(B) \\ {\scriptstyle \vartheta_A}\downarrow & & \downarrow{\scriptstyle \vartheta_B} \\ G(A) & \xrightarrow{G(f)} & G(B) \end{array}$$

Definition 1.5 (Equivalence of Categories)**.**
Two categories $\mathcal{C}, \mathcal{D}$ are **equivalent**, if there are functors $F : \mathcal{C} \longrightarrow \mathcal{D}$, $G : \mathcal{D} \longrightarrow \mathcal{C}$ and two natural isomorphisms $\eta : 1_{\mathcal{C}} \Longrightarrow GF$ $\vartheta : FG \Longrightarrow 1_{\mathcal{D}}$, where $1_{\mathcal{C}}$,$1_{\mathcal{D}}$ are the identity functors on $\mathcal{C}$,$\mathcal{D}$ respectively.

Definition 1.6 ((Cartesian) Product of Categories)**.**
Given two categories $\mathcal{C}$ and $\mathcal{C}'$ their **(cartesian) product** product $\mathcal{C} \times \mathcal{C}'$ a category such that

- objects are pairs (A, A') where A, A' are objects of $\mathcal{C}$, $\mathcal{C}'$ respectively
- a morphism $h : (A, A') \longrightarrow (B, B')$ is a pair $h = (f, f')$ such that $f : A \longrightarrow B$ and $f' : A' \longrightarrow B'$ are arrows in $\mathcal{C}$ and $\mathcal{C}'$ respectively

- composition and identities are defined componentwise i.e. for any $(g, g') : (B, B') \longrightarrow (C, C')$ $(g, g') \circ (f, f') = (g \circ f, g' \circ f')$ and $1_{(A,A')} = (1_A, 1_{A'})$.

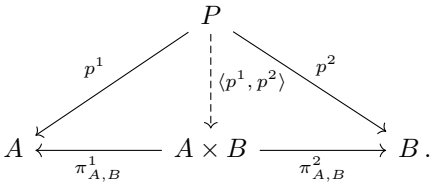
Definition 1.7 ((Cartesian) Products)**.** Let $\mathcal{C}$ be a category. Then a **product diagram** for the objects A,B consists of an object, usually denoted by $A \times B$, and arrows

$$A \xleftarrow[\pi^1_{A,B}]{} A \times B \xrightarrow[\pi^2_{A,B}]{} B$$

such that for any other diagram

$$A \xleftarrow[p^1]{} P \xrightarrow[p^2]{} B$$

there is a unique arrow, usually denoted by $\langle p^1, p^2 \rangle : P \longrightarrow A \times B$, which makes the following diagram commute



The uniqueness-condition is equivalent to the following equation holding for all $h : P \longrightarrow A \times B$:

$$\langle \pi^1_{A,B} \circ h, \pi^2_{A,B} \circ h \rangle = h$$

We then call $A \times B$ the **(cartesian) product** of A and B. It is commutative and associative. We say $\mathcal{C}$ has products if it contains products of all pairs of objects. A category which has (binary) products also has products between any higher amount of objects, e.g. trinary products $A \times B \times C$ for objects A, B, C (parenthesis may be dropped due to associativity).

Definition 1.8 (Terminal Object)**.** An object T of a category $\mathcal{C}$ is called **terminal** if for all objects A there is a unique arrow

$$\nabla_A : A \longrightarrow T\,.$$

Terminal objects are unique up to isomorphism, they can be thought of as 0-ary products.

Definition 1.9 (Finite Products, Cartesian Categories)**.** Let $\mathcal{C}$ be a category. We say $\mathcal{C}$ has **finite products** or $\mathcal{C}$ is **cartesian** if it has products and a terminal object.

Further, more advanced, notions of category theory will be introduced during the following chapters of this thesis.

1.4 Introduction to Quantum Computation

1.4.1 Qubits and States

The basic unit of information in quantum computers is a so called *qubit*. A qubit is a two-level quantum system. Since quantum mechanics is usually modeled in

(complex) Hilbert spaces, a qubit is modeled by a two-dimensional (complex) Hilbert space $\mathcal{H}$, say $\mathcal{H} = \mathbb{C}^2$. Each state the qubit can take, corresponds to a normed vector in $\mathcal{H}$. The normality condition is necessary so that scalar products of states can be interpreted as probabilities.

Theorem 1.10 (Riesz Representation Theorem)**.** *Let H be a Hilbert space and $H^* = \{f : H \longrightarrow \mathbb{C} | f$ linear and bounded$\}$ its dual space. Then the map*

$$\begin{aligned} R_H : H &\longrightarrow H^* \\ \varphi &\longmapsto \langle\varphi|\cdot\rangle \end{aligned} \tag{1.1}$$

is an antilinear isometric bijection. We call R_H^{-1} the Riesz representation. □

This theorem justifies the following notation.

Notation 1.11. We write a state/vector in our Hilbert space as $|\varphi\rangle$, and the functional it represents in the dual space (cmp. theorem 1.10) as $\langle\varphi|$. Then for any φ, ψ we get that

$$\langle\varphi| \left(|\psi\rangle\right) = \langle\varphi|\psi\rangle \tag{1.2}$$

where $\langle\cdot|\cdot\rangle$ is the scalar product in $\mathcal{H}$ justifying our notation. A dual vector is also called a **bra** and a "normal" vector a **ket**, so that combined they form the word *bra-c-ket*, which is an alternative name for the scalar product.

Example 1.12. In the $\mathbb{C}^n$ with a fixed basis and the standard scalar product a state is a column vector. By linearity every functional f must be of the form

$$f(z_1, .., z_n) = \sum_{i=1}^{n} \overline{w}_i z_i = \begin{pmatrix} \overline{w}_1 & ... & \overline{w}_n \end{pmatrix} \begin{pmatrix} z_1 \\ . \\ . \\ z_n \end{pmatrix} \tag{1.3}$$

for some constants $w_i \in \mathbb{C}$; the overline denotes complex conjugation. The latter is costumary here, complex conjugation can of course be avoided by simply redefining the constants, but keeping it suits us well. The functional can thus be represented by some row-vector $\begin{pmatrix} w_1 & ... & w_n \end{pmatrix}$. So while in this case the kets are column-vectors, the bras correspond to conjugated row-vectors. Note that there is an obvious antilinear bijection between them. Applying the latter to the former results in the scalar product

$$\begin{pmatrix} \overline{w}_1 & ... & \overline{w}_n \end{pmatrix} \begin{pmatrix} z_1 \\ . \\ . \\ z_n \end{pmatrix} = \langle \begin{pmatrix} w_1 \\ . \\ . \\ w_n \end{pmatrix} | \begin{pmatrix} z_1 \\ . \\ . \\ z_n \end{pmatrix} \rangle_{\mathbb{C}^n}$$

The Riesz representation theorem really just states, that what we observed above generalizes to all Hilbert spaces. The Dirac notation is a clever way to generalize the notation, the row vectors become *bras* and the column vectors *kets*.

The standard basis of $\mathcal{H}$ is usually denoted $|0\rangle$, $|1\rangle$, physically $|0\rangle$ is the *ground state* and $|1\rangle$ the *excited state* of the system. Each $|\psi\rangle$ can thus be

written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad \text{with} \qquad |\alpha|^2 + |\beta|^2 = 1\,,$$

where $|\alpha|^2$ and $|\beta|^2$ are the probabilities, that $|\psi\rangle$ will, when measured, be found in state $|0\rangle$ and $|1\rangle$ respectively.

Definition 1.13 (Tensor product of Hilbert spaces)**.** Let H_1, H_2 be Hilbert spaces with bases $a_1, .., a_n$ and $b_1, .., b_k$ respectively, then $H_1 \otimes H_2$ is the $n \cdot k$-dimensional vector space spanned by the pairs of bases vectors $a_i \otimes b_j$. To make it a Hilbert space, an inner product can be defined the following way: $\forall \psi_1, \varphi_1 \in H_1\ \forall \psi_2, \varphi_2 \in H_2$

$$\langle \psi_1 \otimes \psi_2 | \varphi_1 \otimes \varphi_2 \rangle_{H_1 \otimes H_2} := \langle \psi_1 | \varphi_1 \rangle_{H_1} \cdot \langle \psi_2 | \varphi_2 \rangle_{H_2}\,. \tag{1.4}$$

The state of n qubits is a normalized vector in the Hilbert space $\mathcal{H}^{\otimes n} := \mathcal{H} \otimes .. \otimes \mathcal{H}$, where the tensor product is taken $n-1$ times. To shorten the notation we write $|q_1, .., q_n\rangle$ for $|q_1\rangle \otimes .. \otimes |q_n\rangle$, where $q_1, .., q_n \in \{0,1\}$. The canonical basis for $\mathcal{H}^{\otimes n}$ is then

$$\big\{\, |q_1, .., q_n\rangle \,\big|\, q_1, .., q_n \in \{0,1\}\big\}. \tag{1.5}$$

Note that in a two-particle system there are states, which can not be represented as the product of two one-particle states. Take for example the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle,$$

then there are no single particle states $|\varphi_1\rangle, |\varphi_2\rangle$ with $|\psi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle$. These particles are then called *entangled*. Analogously entanglement is possible for more than two particles.

1.4.2 Operators, Observables and Gates

Operators and their Representations

Given a finite dimensional Hilbert space $\mathcal{H}$, an operator O on $\mathcal{H}$ is a linear map $O : \mathcal{H} \longrightarrow \mathcal{H}$. Fixing a basis $\{\varphi_i\}_{i=1,..,n}$ of $\mathcal{H}$ every operator can be written as a matrix, i.e. there are $\{z_{ij}\}_{i,j=1,..,n} \in \mathbb{C}$ such that

$$O = \begin{pmatrix} z_{11} & ... & z_{1n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ z_{n1} & ... & z_{nn} \end{pmatrix} = \sum_{i,j=1}^{n} z_{ij} |\varphi_i\rangle\langle\varphi_j|\,. \tag{1.6}$$

Observables and self-adjoint operators

In quantum mechanics in a system described by a Hilbert space $\mathcal{H}$ an observable is modeled by a self-adjoint operator of $\mathcal{H}$, i.e. an operator $A : \mathcal{H} \longrightarrow H$ s.t.

$$\forall \psi, \psi' \in \mathcal{H}\ \langle \psi' | A\psi \rangle = \langle A\psi' | \psi \rangle\,. \tag{1.7}$$

The possible numerical outcomes of a measurement of that observable corresponds to the eigenvalues of A.

Theorem 1.14 (Spectral theorem)**.** *Let $\mathcal{H}$ be a finite dimensional Hilbert space and $A : \mathcal{H} \longrightarrow \mathcal{H}$ a self-adjoint operator. Then there is an orthonormal basis of $\mathcal{H}$, that consists of eigenvectors/eigenstates of A.* □

Measurement

Let a physical system be described by a finite dimensional Hilbert space $\mathcal{H}$. Let A be an observable of that system and $|\Psi\rangle \in \mathcal{H}$ its current state. We denote by $|\varphi_i\rangle_{i=1,..,n}$ an orthonormal basis of $\mathcal{H}$ consisting of eigenvectors of A and by $(a_i)_{i=1,..,n}$ the corresponding eigenvalues, i.e.

$$\forall i, j = 1, .., n \qquad A|\varphi_i\rangle = a_i|\varphi_i\rangle \qquad \text{and} \qquad \langle\varphi_j|\varphi_i\rangle = \delta_{ij}\,. \tag{1.8}$$

Then the expected value of the outcome of a measurement is given by $\langle\Psi|A\Psi\rangle$. Expanding $|\Psi\rangle = \sum_{i=1}^{n}\langle\varphi_i|\Psi\rangle|\varphi_i\rangle$ we obtain

$$\begin{aligned}\langle\Psi|A\Psi\rangle &= \sum_{i,j=1}^{n}\langle\Psi|\varphi_j\rangle\langle\varphi_j|A\varphi_i\rangle\langle\varphi_i|\Psi\rangle = \sum_{i,j=1}^{n}\langle\Psi|\varphi_j\rangle a_i\langle\varphi_j|\varphi_i\rangle\langle\varphi_i|\Psi\rangle \\ &= \sum_{i=1}^{n}|\langle\Psi|\varphi_i\rangle|^2 a_i\end{aligned}$$

In the expanded form $|\langle\Psi|\varphi_i\rangle|^2$ is the probability to find the system in state φ_i while a_i is the value the observable has in this state. During the measurement of A the system collapses to an eigenstate

$$|\Psi\rangle \longmapsto |\varphi_m\rangle$$

whos eigenvalue a_m is the outcome of the measurement. The system then remains in that eigenstate even after subsequent measurements of A, measurements of other observables however may well change the state again. The exact nature of this collaps is not well understood and will not be further explored here.

Gates and Unitary Transformations

Excepting the extraordinary case of the measurement, a state can only be transformed using unitary transformations, i.e. operators $U : \mathcal{H} \longrightarrow \mathcal{H}$ s.t.

$$\forall\varphi, \psi \in \mathcal{H} \;\; \langle U\varphi|U\psi\rangle = \langle\varphi|\psi\rangle\,. \tag{1.9}$$

The unitarity is necessary for preservation of normality of the states. Thus in a quantum computer instead of logical gates we have so called unitary gates corresponding to unitary transformations. The most prominent of those trans-

formations are, in our 2-dimensional case, the Pauli matrices.

$$\begin{aligned}\sigma_x := \sigma_1 &:= \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} &&= |1\rangle\langle 0| + |0\rangle\langle 1|\\ \sigma_y := \sigma_2 &:= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} &&= i|1\rangle\langle 0| - i|0\rangle\langle 1|\\ \sigma_z := \sigma_3 &:= \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} &&= |0\rangle\langle 0| - |1\rangle\langle 1|\end{aligned} \tag{1.10}$$

The Pauli matrices are not only unitary, but also self adjoint, thus they can also be interpreted as observables. Another very important unitary transformation which can be realized as a gate is the Hadamard transformation

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\langle 0| + \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\langle 1|\,. \tag{1.11}$$

Applying it to a basis vector produces an entangled state, which can be found to be $|0\rangle$ or $|1\rangle$ with equal probability.

1.4.3 No Cloning and No Deletion

We investigate whether a *cloning* operator can exist, that is an operator that copies the state of a qubit onto another qubit. More precisely, is there a unitary transformation $C : \mathcal{H}\otimes\mathcal{H} \longrightarrow \mathcal{H}\otimes\mathcal{H}$ such that for some state $|\chi\rangle$

$$C\left(|\varphi\rangle \otimes |\chi\rangle\right) = |\varphi\rangle \otimes |\varphi\rangle \quad \forall\varphi \in \mathcal{H}\,? \tag{1.12}$$

To answer this question let U be an arbitrary unitay transformation, we want to show that $C \neq U$. For any two distinct, non-orthogonal, states $|\varphi\rangle, |\psi\rangle$ we have that

$$\langle C\left(\varphi\otimes\chi\right) | \, C\left(\psi\otimes\chi\right)\rangle = \langle\varphi\otimes\varphi|\psi\otimes\psi\rangle = \langle\varphi|\psi\rangle^2$$

on the other hand

$$\langle U\left(\varphi\otimes\chi\right) | \, U\left(\psi\otimes\chi\right)\rangle = \langle\varphi\otimes\chi|\psi\otimes\chi\rangle = \langle\varphi|\psi\rangle\langle\chi|\chi\rangle = \langle\varphi|\psi\rangle\,.$$

Since $|\varphi\rangle, |\psi\rangle$ are distinct and non-orthogonal we have $0 \lneq \langle\varphi|\psi\rangle \lneq 1$ and thus also

$$\langle C\left(\varphi\otimes\chi\right) | \, C\left(\psi\otimes\chi\right)\rangle = \langle\varphi|\psi\rangle^2 \neq \langle\varphi|\psi\rangle = \langle U\left(\varphi\otimes\chi\right) | \, U\left(\psi\otimes\chi\right)\rangle \tag{1.13}$$

proving that $C \neq U$.

Since all unitary transformations are invertible, the above also shows, that there is no *deletion* operator, i.e. there is no unitary $D : \mathcal{H}\otimes\mathcal{H} \longrightarrow \mathcal{H}\otimes\mathcal{H}$ such that for some $|\chi\rangle$

$$D\left(|\varphi\rangle \otimes |\varphi\rangle\right) = |\varphi\rangle \otimes |\chi\rangle \quad \forall\varphi \in \mathcal{H}\,. \tag{1.14}$$

1.4.4 Advantages and Limitations

A number of algorithms for quantum computers have been found for solving problems whose solution is considerably more complex or even unfeasable in a classical setting. The most prominent of those are *Shor's algorithm* [24] solving the *hidden subgroup problem* for finite abelian groups, special cases of which include finding prime factors of an integer and computing discrete logarithms and *Grover's algorithm* [14] for database search. Quantum computers do however also come with disadvantages and limitations. Due to the probabilistic nature of the quantum realm, the mentioned algorithms may with a certain probability yield an incorrect result. As a consequence calculations have to be carried through repeatedly to reduce the probability of an error. This entails, that quantum computers are actually slower at solving tasks at which the quantum effects do not provide an algorithmical advantage. It should also be noted, that any quantum computer additionally contains a classical computing unit, which controls the qubits.

Chapter 2

The Internal Logic of Quantum Mechanics

In this chapter, we derive a logical system which can describe the inner workings of quantum mechanics. To be able to transverse from physics to logic, we analyze in the first section a mathematical model of quantum mechanics, namely the theory of Hilbert spaces, categorically, and then build a logical system which corresponds to the internal logic of that category. This approach was pioneered among others by J.C. Baez [6].

2.1 A Hilbert Space Category

2.1.1 Morphisms of Hilbert Spaces

The world of Quantum Mechanics is mathematically usually modeled in Hilbert Spaces. We thus aim towards building a category **Hilb** of Hilbert spaces i.e. a category which has Hilbert spaces as objects. But what should the morphisms be? Remember, that a (complex) Hilbert Space H is a $\mathbb{C}$-vector space endowed with an inner product $\langle \cdot \,|\, \cdot \rangle$. An obvious choice for a morphism $f : H \longrightarrow H'$ in **Hilb** would thus be a linear function/operator $f : H \longrightarrow H'$ which is also an isometry, i.e. it respects the inner product structure:

$$\langle f(\psi) | f(\varphi) \rangle_{H'} = \langle \psi | \varphi \rangle_H \qquad \forall \varphi, \psi \in H \,.$$

While this certainly makes for an interesting category, for the description of quantum mechanics it has turned out to be too restrictive as not all operators in use need to be isometries (cmp. section 1.4.2). Instead, we define the category in the following way:

Definition 2.1 (Hilbert Space Category)**.**
Hilb[1] is the category which has as
objects: finite dimensional Hilbert spaces H, H'
arrows: bounded linear functions $f : H \longrightarrow H'$

[1] It should be noted that in the literature this category is often called **finHilb** or similiar, to distinguish it from categories which also include infinite-dimensional Hilbert spaces. Since the latter is of no importance in this thesis, we stick with the simpler **Hilb**.

where *bounded* is meant in the usual way:

$$\exists_{M\in\mathbb{R}}\ \forall_{\psi\in H}\ :\ \langle f(\psi)|f(\psi)\rangle_{H'} \leq M\langle\psi|\psi\rangle_H\,.$$

It is worth noting that in our finite-dimensional setting boundedness is already a consequence of linearity.

In the way we defined **Hilb**, we can show that it is equivalent to the category **Vec** of finite-dimensional (complex) vector spaces and linear maps. Indeed, if U : **Hilb** $\longrightarrow$ **Vec** is the corresponding forgetful functor we can show the following lemma.

Lemma 2.2. *Let H, H' be two objects of* **Hilb***, then*

$$H \cong H' \qquad \Longleftrightarrow \qquad U(H) \cong U(H')\,.$$

In particular, if V is an object of **Vec** *and $\langle\cdot\,|\cdot\rangle_1$, $\langle\cdot\,|\cdot\rangle_2$ two scalar products on V, then*

$$(V, \langle\cdot\,|\cdot\rangle_1) \cong (V, \langle\cdot\,|\cdot\rangle_2) \qquad\qquad \textit{in } \textbf{Hilb}.$$

Proof. $\Rightarrow$ is immediate. For the $\Leftarrow$-implication note that any isomorphism $f : U(H) \longrightarrow U(H')$ is automatically also an isomorphism $f : H \longrightarrow H'$ in **Hilb**, since by our definition the additional structure of Hilbert spaces need not be respected by arrows in **Hilb**. □

Corollary 2.3. **Hilb** *and* **Vec** *are equivalent.*

Proof. Let T be a functor T : **Vec** $\longrightarrow$ **Hilb**, $V \mapsto (V, \langle\cdot|\cdot\rangle)$ where $\langle\cdot|\cdot\rangle$ is an arbitrary scalar product on V. This is ambiguous but any such functor will do. By definiton $UT = id_{\mathbf{Vec}}$, so it remains to show that there is a natural transformation $\vartheta : TU \longrightarrow id_{\mathbf{Hilb}}$. But using Lemma 2.2 we have an $\vartheta_H : TU(H) \xrightarrow{\cong} H$ for any H in **Hilb**. We need to show that the following diagram commutes:

$$\begin{array}{ccc} TU(H) & \xrightarrow{TU(f)} & TU(H') \\ {\scriptstyle\vartheta_H}\downarrow & & \downarrow{\scriptstyle\vartheta_{H'}} \\ H & \xrightarrow{\quad f\quad} & H' \end{array}$$

However this is trivial since the $\vartheta_H, \vartheta_{H'}$ are induced by the identities on the underlying vector spaces, and T can be chosen such that TU acts as the identity on arrows, since any linear maps is bounded in our setting. □

Thus at first glance our category of choice seems senseless, why not just choose **Vec**? And while some of the arguments made depend on finiteness of dimension, a similiar issue arises even when including infinite-dimensional spaces [5]. Nevertheless in the next section we will be able to show that the Hilbert space structure is essential and **Hilb** not replacable by a simple vector space category.

2.1.2 Hilb as a †-Category

Firstly we introduce †-categories (pronounced: dagger categories).

Definition 2.4 (†-category)**.** A †-category (or **dagger** category) $(\mathcal{C},\dagger)$ is a category $\mathcal{C}$ together with an endofunctor $\dagger : \mathcal{C} \longrightarrow \mathcal{C}$ such that

$$A^\dagger := \dagger(A) = A \qquad \text{on objects} \tag{2.1}$$

and for objects A, B

$$\dagger : \hom_{\mathcal{C}}(A,B) \longrightarrow \hom_{\mathcal{C}}(B,A) \tag{2.2}$$
$$f \longmapsto f^\dagger \tag{2.3}$$

such that the following three identities hold.

$$id_A^\dagger = id_A\,, \tag{2.4}$$
$$f^{\dagger\dagger} = f\,, \tag{2.5}$$
$$(g \circ f)^\dagger = f^\dagger \circ g^\dagger\,, \tag{2.6}$$

for any arrow $g : B \longrightarrow C$.

We now construct a †-endofunctor $\dagger : \mathbf{Hilb} \longrightarrow \mathbf{Hilb}$ using inner products. For a morphism $f : H \longrightarrow H'$ we define $f^\dagger$ s.t. for all $\psi \in H$ and $\varphi \in H'$

$$\langle f(\psi)|\varphi\rangle_{H'} = \langle \psi|f^\dagger(\varphi)\rangle_H\,.$$

It can be shown, that this is a well defined map using the property

$$x = y \qquad \Longleftrightarrow \qquad \langle x|z\rangle = \langle y|z\rangle \;\; \forall z \in H \tag{2.7}$$

which holds for general Hilbert spaces H. From that we can also conclude

$$\langle id_H^\dagger(\psi)|\varphi\rangle_H = \langle \psi|id_H(\varphi)\rangle_H = \langle\psi|\varphi\rangle_H = \langle id_H(\psi)|\varphi\rangle_H \quad \text{and thus} \quad id_H^\dagger = id_H$$
$$\langle f(\psi)|\varphi\rangle_{H'} = \langle\psi|f^\dagger(\varphi)\rangle_H = \langle f^{\dagger\dagger}(\psi)|\varphi\rangle_{H'} \quad \text{and thus} \quad f = f^{\dagger\dagger}$$

$$\langle\psi|(g\circ f)^\dagger(\varphi)\rangle_H = \langle g(f(\psi))|\varphi\rangle_{H''} = \langle f(\psi)|g^\dagger(\varphi)\rangle_{H'} = \langle\psi|f^\dagger(g^\dagger(\varphi))\rangle_H$$
$$\text{and thus } (g\circ f)^\dagger = g^\dagger \circ f^\dagger\,.$$

Thus as such $(\mathbf{Hilb},\dagger)$ becomes a †-category. Note that the use of inner products of the domain as well as of the codomain were absolutely crucial to defining †. This dependency even goes both ways, meaning the following:

Lemma 2.5. *Let $U_\dagger : (\mathbf{Hilb},\dagger) \longrightarrow (\mathbf{Vec},\dagger)$ be the functor of †-categories canonically induced by the forgetful functor U. Then $U_\dagger$ is invertible i.e. all inner products can be recovered in $(\mathbf{Vec},\dagger)$.*

Proof. It is sufficient to construct an inner product for each vector space using the functor †, the rest follows immediately. For that let first H be an object of **Hilb**. There is a natural bijection between elements $\psi \in H$ and operators

$T : \mathbb{C} \longrightarrow H$ given by

$$\begin{aligned} H &\longrightarrow (\mathbb{C} \longrightarrow H) \\ \psi &\longmapsto (T_\psi : z \mapsto z\psi) \end{aligned}$$

We can then note the following identity.

$$\langle \psi | \varphi \rangle = \left(T_\psi^\dagger \circ T_\varphi \right)(1) \,. \tag{2.8}$$

Indeed

$$\langle \psi | \varphi \rangle_H = \langle T_\psi(1) | T_\varphi(1) \rangle_H = \langle 1 | T_\psi^\dagger \left(T_\varphi(1) \right) \rangle_{\mathbb{C}} = \left(T_\psi^\dagger \circ T_\varphi \right)(1) \,.$$

Since $U_\dagger$ preserves all morphisms we can reconstruct the scalar product on $U_\dagger(H)$ by setting

$$\langle U_\dagger(\psi) | U_\dagger(\varphi) \rangle_{U_\dagger(H)} = \left(U_\dagger(T_\psi^\dagger) \circ U_\dagger \left(T_\varphi \right) \right)(1) \,.$$

Making $U_\dagger(H)$ into a Hilbert space which per definition coincides with H. □

At this point it should be noted, that this works only for the specific functor $\dagger$. Generally in a different $\dagger$-category $(\mathbf{Vec}, \dagger')$ no scalar product can be constructed from the functor $\dagger'$, which affirms the necessity of the Hilbert-space structure.

2.1.3 Monoidal Categories

Another important structure in quantum mechanics is the tensor product. In can be captured categorically with the following notion.

Definition 2.6 (Monoidal Category)**.** A **monoidal** category $(\mathcal{C}, \otimes, I, a, l, r)$ is a category $\mathcal{C}$ together with a functor $\otimes : \mathcal{C} \times \mathcal{C} \longrightarrow \mathcal{C}$ and

- a **unit** object I of $\mathcal{C}$
- a natural isomorphism called the **associator**, which to each three objects X, Y, Z assigns an isomorphism

$$a_{X,Y,Z} : (X \otimes Y) \otimes Z \xrightarrow{\sim} X \otimes (Y \otimes Z)$$

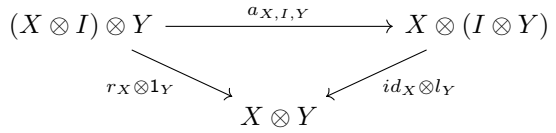
- two natural isomorphisms called **left** and **right unitors**, which assign to each object X the isomorphisms

$$l_X : I \otimes X \longrightarrow X \qquad \text{and} \qquad r_X : X \otimes I \longrightarrow X$$

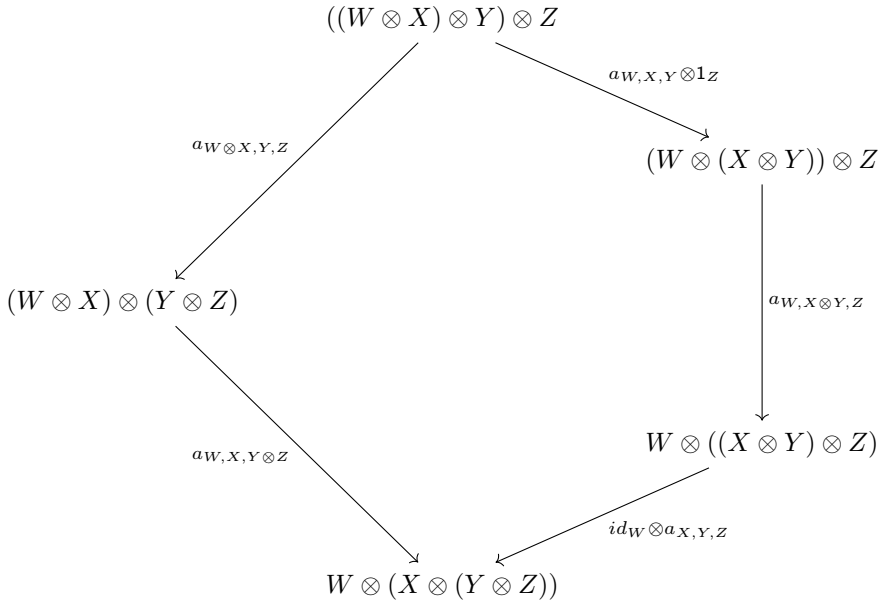
respectively,

such that

- for all objects X,Y the triangle equation holds:



- for all objects W, X, Y, Z the pentagon equation holds:



When the rest of the data is irrelevant or clear from context, we only write $(\mathcal{C}, \otimes)$ or $\mathcal{C}$ for a monoidal category.

Let's analyze similarities and differences between monoidal categories and categories with products.

Lemma 2.7. *A category with finite (cartesian) products (and terminal object T) is monoidal.*

Proof. As tensor product we can take the cartesian product, T then acts as the unit object. The rest of the proof is straightfoward. □

One major difference between cartesian and tensor products is the existence (or non-existence) of the so called duplication and deletion morphisms. In every category with finite cartesian products one may define the morphisms

$$\textit{duplication}: \qquad \Delta_X : X \longrightarrow X \times X \tag{2.9}$$

$$\textit{deletion}: \qquad \nabla_X : X \longrightarrow T\,. \tag{2.10}$$

this need not be possible in monoidal categories. Indeed in **Hilb** with the standard tensor product it turns out to be impossible.

Example 2.8. Hilb becomes a monoidal category when endowed with the usual tensor product. As unit object we take the 1-dimensional Hilbert space $\mathbb{C}$ itself.

Definition 2.9 (Braided monoidal category)**.**
A **braided monoidal** category $(\mathcal{C}, b)$ is a monoidal category $\mathcal{C}$ together with a natural isomorphism b called braiding, which assigns to objects X, Y an isomorphism

$$b_{X,Y} : X \otimes Y \longrightarrow Y \otimes X$$

s.t. the hexagon-equations hold:

$$\begin{array}{ccccc} X \otimes (Y \otimes Z) & \xrightarrow{a^{-1}_{X,Y,Z}} & (X \otimes Y) \otimes Z & \xrightarrow{b_{X,Y} \otimes 1_Z} & (Y \otimes X) \otimes Z \\ \downarrow{\scriptstyle b_{X,Y\otimes Z}} & & & & \downarrow{\scriptstyle a_{Y,X,Z}} \\ (Y \otimes Z) \otimes X & \xleftarrow[a^{-1}_{Y,Z,X}]{} & Y \otimes (Z \otimes X) & \xleftarrow[1_Y \otimes b_{X,Z}]{} & Y \otimes (X \otimes Z) \end{array}$$

$$\begin{array}{ccccc} (X \otimes Y) \otimes Z & \xrightarrow{a_{X,Y,Z}} & X \otimes (Y \otimes Z) & \xrightarrow{1_X \otimes b_{Y,Z}} & X \otimes (Z \otimes Y) \\ \downarrow{\scriptstyle b_{X\otimes Y,Z}} & & & & \downarrow{\scriptstyle a^{-1}_{X,Z,Y}} \\ Z \otimes (X \otimes Y) & \xleftarrow[a_{Z,X,Y}]{} & (Z \otimes X) \otimes Y & \xleftarrow[b_{X,Z} \otimes 1_Y]{} & (X \otimes Z) \otimes Y \end{array}$$

The hexagon-equations ensure what one might call *uniqueness of canonicity*. That means if the structure provided yields two canonical ways of achieving something, then these two should coincide. In the first hexagon equation the structure provides two paths from $X \otimes (Y \otimes Z)$ to $(Y \otimes Z) \otimes X$, and they should thus coincide. This motive continues throughout this thesis.

Definition 2.10 (Symmetric monoidal category)**.**
A **symmetric monoidal** category is a braided monoidal category $\mathcal{C}$ such that for all objects X, Y

$$b^{-1}_{X,Y} = b_{Y,X} \,,$$

where $b_{X,Y}$ is the braiding.

We already mentioned that one difference between cartesian and monoidal categories is the existence of duplication and deletion morphisms. For symmetric monoidal categories this turns out to be the only difference, in the following sense

Theorem 2.11. *Let $(\mathcal{C}, \otimes, I, a, l, r, b)$ be a symmetric monoidal category. Then $(\mathcal{C}, \otimes, I)$ is cartesian if and only if for every object X there is a duplication and a deletion morphism as defined in (2.9) and (2.10) which respect the symmetric monoidal structure.*

This is purposefully stated vaguely to avoid overloading the reader with even more diagrams. Generally what we mean is that diagrams containing the duplication and deletion arrows as well as the morphisms that the symmetric monoidal structure yields commute. For example we demand $(f \otimes f) \circ \Delta_Z = \Delta_X \circ f$ for any arrow $f : Z \longrightarrow X$.

Proof. "⇒": Let $\mathcal{C}$ be cartesian. Then since I is terminal we have deletion $\nabla_X : X \longrightarrow I$ given for every X. Duplication can be defined through the

pairing of identities: $\Delta_X = \langle id_X, id_X \rangle$.
"$\Leftarrow$": Let there be deletion and duplication. Then the deletion morphisms render I terminal. Uniqueness of the morphisms follows from *respecting the structure*, in particular we have $\nabla_I = id_I$ by this demand. The pairing and projections for objects X, Y and arrows $f : Z \longrightarrow X$, $g : Z \longrightarrow Y$ can be constructed as follows:

$$\begin{aligned}
\pi^1_{X,Y} &:= r_X \circ (id_X \otimes \nabla_Y) \\
\pi^2_{X,Y} &:= l_Y \circ (\nabla_X \otimes id_Y) \\
\langle f, g \rangle &:= (f \otimes g) \circ \Delta_Z \,.
\end{aligned}$$

For an arrow $h : Z \longrightarrow X \otimes Y$

$$\begin{aligned}
\langle \pi^1_{X,Y} \circ h, \pi^2_{X,Y} \circ h \rangle &= \big(\big(r_X \circ (id_X \otimes \nabla_Y) \circ h \big) \otimes \big(l_Y \circ (\nabla_X \otimes id_Y) \circ h \big) \big) \circ \Delta_Z \\
&= \big(\big(r_X \circ (id_X \otimes \nabla_Y) \big) \otimes \big(l_Y \circ (\nabla_X \otimes id_Y) \big) \big) \circ (h \otimes h) \circ \Delta_Z \\
&= \big(\big(r_X \circ (id_X \otimes \nabla_Y) \big) \otimes \big(l_Y \circ (\nabla_X \otimes id_Y) \big) \big) \circ \Delta_{X \otimes Y} \circ h
\end{aligned}$$

So it remains to show $\big(\big(r_X \circ (id_X \otimes \nabla_Y) \big) \otimes \big(l_Y \circ (\nabla_X \otimes id_Y) \big) \big) \circ \Delta_{X \otimes Y} = id_{X \otimes Y}$. This amounts to saying that the following diagram commutes

$$\begin{array}{ccc}
X \otimes Y & \xrightarrow{\Delta_{X \otimes Y}} & (X \otimes Y) \otimes (X \otimes Y) \\
{\scriptstyle id_{X \otimes Y}} \downarrow & & \downarrow {\scriptstyle (1_X \otimes \nabla_Y) \otimes (\nabla_X \otimes 1_Y)} \\
X \otimes Y & \xleftarrow[r_X \otimes l_Y]{} & (X \otimes I) \otimes (I \otimes Y)
\end{array}$$

which follows from the *respecting the structure*-demand and the properties of symmetric monoidal categories we've seen above. More details can be found in the literature [21]. □

2.1.4 Closed Categories

Definition 2.12 (Closed categories)**.**
A monoidal category $\mathcal{C}$ is called **left closed**, if there is an **internal hom** functor

$$\multimap : \mathcal{C}^{op} \times \mathcal{C} \longrightarrow \mathcal{C}$$

together with a natural isomorphism, that assigns to each triple of objects X, Y, Z a bijection

$$c_{X,Y,Z} : \hom(X \otimes Y, Z) \longrightarrow \hom(X, Y \multimap Z)$$

and it is called **right closed** if accordingly

$$c_{X,Y,Z} : \hom(X \otimes Y, Z) \longrightarrow \hom(Y, X \multimap Z).$$

We call the action of c *currying*.

In the following we only consider right closed categories. Note, that in the case of braided monoidal categories this makes no difference since $X \otimes Y \cong Y \otimes X$ by the braiding.

Example 2.13. The category **Hilb** can be made into a closed category by defining $X \multimap Y$ as the space of bounded linear operators $f, g : X \longrightarrow Y$ with the standard scalar product:

$$\langle f|g\rangle_{X\multimap Y} := \sum_{i=1}^{n} \langle f(e_i)|g(e_i)\rangle_Y = \mathrm{Tr}[f^\dagger \circ g] \tag{2.11}$$

where $(e_i)_{i=1,..,n}$ is a basis of X and Tr denotes the trace-operation. For $h : X \otimes Y \longrightarrow Z$, we define $\tilde{h} : Y \longrightarrow (X \multimap Z)$ by $\tilde{h}(y) : x \mapsto h(x \otimes y)$ for any $y \in Y$. Then $c_{X,Y,Z}$ is given by

$$c_{X,Y,Z} : h \longmapsto \tilde{h}\,.$$

2.1.5 Compact Categories

Definition 2.14 (Dual objects)**.**
In a monoidal category $\mathcal{C}$, we call an object X^* the **right dual** of an object X (and accordingly X the **left dual** of X^*) if there are morphisms

$$i_X : I \longrightarrow X^* \otimes X \qquad \text{and} \qquad e_X : X \otimes X^* \longrightarrow I$$

called the **unit** and the **counit** respectively, s.t. the zigzag-equations are satisfied:

$$\begin{array}{ccccc} X \otimes I & \xrightarrow{id_X \otimes i_X} & X \otimes (X^* \otimes X) & \xrightarrow{a^{-1}_{X,X^*,X}} & (X \otimes X^*) \otimes X \\ {\scriptstyle r_X}\downarrow & & & & \downarrow{\scriptstyle e_X \otimes id_X} \\ X & \xleftarrow[l_X]{} & & & I \otimes X \end{array}$$

$$\begin{array}{ccccc} I \otimes X^* & \xrightarrow{i_X \otimes id_{X^*}} & (X^* \otimes X) \otimes X^* & \xrightarrow{a_{X^*,X,X^*}} & X^* \otimes (X \otimes X^*) \\ {\scriptstyle l_X}\downarrow & & & & \downarrow{\scriptstyle id_{X^*} \otimes e_X} \\ X^* & \xleftarrow[r_X]{} & & & X^* \otimes I \end{array}$$

Note that in the case of a symmetric monoidal category the notions of right and left dual coincide.

In a closed symmetric monoidal category $\mathcal{C}$ a candidate for a dual object of X is always $X \multimap I$. For this so called *weak dual* we can find the counit

$$e_X := c^{-1}_{X,X\multimap I,I}(id_{X\multimap I})\,. \tag{2.12}$$

where c is currying. It acts as follows on elements $x \in X, \varphi \in X^*$

$$e_X(x \otimes \varphi) = \varphi(x)$$

and can thus also be called an *evaluation morphism*. A unit need not always exist.

Example 2.15. For the category **Hilb** we set $X^* := X \multimap \mathbb{C}$. The counit exists by above considerations. To find a unit we first note that there is a

natural isomorphism which assigns for each $X, Y \in \mathbf{Hilb}$

$$j : X^* \otimes Y \longrightarrow X \multimap Y$$

where for any $\varphi \in X^*, y \in Y, f \in X \multimap Y$

$$\begin{aligned} j(\varphi \otimes y)(x) &:= \varphi(x)y \qquad\qquad \forall_{x \in X} \\ j^{-1}(f) &:= \sum_i e_i^* \otimes f(e_i) \end{aligned}$$

where $\{e_i\}_i$ is a basis of X and $\{e_i^*\}_i$ the dual basis. This justifies defining a morphism $j^{-1} \circ i_X : \mathbb{C} \longrightarrow X \multimap X$ instead of the unit. We set

$$\left(j^{-1} \circ i_X\right)(z) := z\, id_X\,.$$

and obtain

$$i_X(z) = j(z\, id_X) = \sum_i e_i^* \otimes z e_i\,.$$

It is straightforward to show that the zigzag-equations are satisfied.

Definition 2.16 (Compact Categories)**.**
A **compact** category is a monoidal category $\mathcal{C}$ such that every object of $\mathcal{C}$ has both a left dual and a right dual.

In case of a compact closed category $\mathcal{C}$ the mapping $X \mapsto X^*$ extends to a contravariant endofunctor, by defining the dual $f^* : Y^* \longrightarrow X^*$ of an arrow $f : X \longrightarrow Y$ of $\mathcal{C}$ to be the unique arrow which makes the following diagram commute:

$$\begin{array}{ccccc} Y^* & \xleftarrow{l_{Y^*}} & I \otimes Y^* & \xrightarrow{i_X \otimes id_{Y^*}} & X^* \otimes X \otimes Y^* \\ {\scriptstyle f^*}\downarrow & & & & \downarrow{\scriptstyle id_{X^*} \otimes f \otimes id_{Y^*}} \\ X^* & \xleftarrow[r_{X^*}]{} & X^* \otimes I & \xleftarrow[id_{X^*} \otimes e_Y]{} & X^* \otimes Y \otimes Y^* \end{array}$$

In equational terms:

$$f^* = r_{X^*} \circ (id_{X^*} \otimes e_Y) \circ (id_{X^*} \otimes f \otimes id_{Y^*}) \circ (i_X \otimes id_{Y^*}) \circ l_{Y^*}^{-1} \tag{2.13}$$

Example 2.17. Hilb is obviously compact. Untangeling the definition, we find, that for an arrow $f : X \longrightarrow Y$ in **Hilb** f^* acts as follows on an element $\psi \in Y^* = Y \multimap \mathbb{C}$:

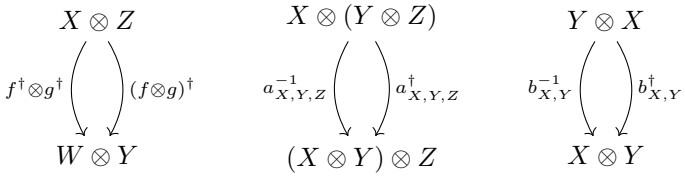
$$f^*(\psi) = \sum_i \psi(f(e_i)) e_i^* = \psi \circ f\,.$$

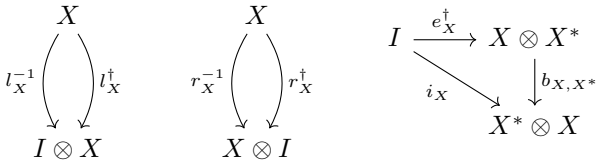
Thus f^* coincides with the commonly used dual map.

2.1.6 Dagger Compact Closed Categories

We now join the concept of a compact closed category with the one of a dagger category.

Definition 2.18 (Dagger compact closed category)**.** A dagger compact closed category is a is a compact closed category $\mathcal{C}$ which is also a †-category such that for all objects W, X, Y, Z and arrows $f : W \longrightarrow X$, $g : Y \longrightarrow Z$ the following diagrams commute





Definition 2.19 (Lower Star Functor)**.** Let $\mathcal{C}$ be a dagger compact closed category and $f : X \longrightarrow Y$ an arrow in $\mathcal{C}$. We then define the covariant lower star functor by $A \mapsto A^*$ and $f \mapsto f_*$, where we define $f_* : X^* \longrightarrow Y^*$ by $f_* := f^{*\dagger} = f^{\dagger *}$.

Example 2.20. Let $f : X \longrightarrow Y$ be an arrow in **Hilb**, $(e_i)_i$ a basis of X and $(e_i^*)_i$ the dual basis. Applying the definition $f_* = f^{*\dagger}$ we find, that for any $\varphi \in X^*$,$\psi \in Y^*$

$$\begin{aligned}\langle f_*(\varphi)|\psi\rangle_{Y^*} = \langle\varphi|f^*(\psi)\rangle_{X^*} = \langle\varphi|\sum_i \psi(f(e_i))e_i^*\rangle_{X^*} = \sum_i \psi(f(e_i))\langle\varphi|e_i^*\rangle_{X^*} \\ = \sum_i \psi(f(e_i))\overline{\varphi(e_i)}\end{aligned}$$

where we used that $\langle\varphi|e_i^*\rangle_{X^*} = \sum_j \langle\varphi(e_j)|e_i^*(e_j)\rangle_{\mathbb{C}} = \overline{\varphi(e_i)}$. The overline denotes complex conjugation. There is a relation between $f, f^\dagger$ and f^*, f_* respectively, namely they make the following diagrams commute

$$\begin{array}{ccc} X & \xleftarrow{f^\dagger} & Y \\ {\scriptstyle R_X}\downarrow & & \downarrow{\scriptstyle R_Y} \\ X^* & \xleftarrow{f^*} & Y^* \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle R_X}\downarrow & & \downarrow{\scriptstyle R_Y} \\ X^* & \xrightarrow{f_*} & Y^* \end{array}$$

where R_X, R_Y are the isometric bijections given by the Riesz representation theorem. Note however that since R_X and R_Y are antilinear they do not belong to the category **Hilb**.

2.2 Linear Logic

In this section we introduce Girard's linear logic [13], or rather a fragment of it.

2.2.1 A Background in Logic and Proof Theory

Classical Logic

Classical logic, more precisely classical propositional calculus, is the theory of abstract propositions $A, B, ..$ and their connectives namely the conjunction $\wedge$ (*and*), the disjunction $\vee$ (*or*) and the implication $\Rightarrow$ together with a *verum*, a notion of truth, usually denoted by $\mathsf{1}$ or $\top$ and a *falsum*, a notion of falsity, denoted by $\mathsf{0}$ or $\bot$.

In proof theory we consider the logical rules spanning proofs. We define a *sequent*

$$A_1, .., A_n \vdash B_1, .., B_m \tag{2.14}$$

to mean *at least one of the propositions* $B_1, .., B_m$ *is provable from propositions* $A_1, .., A_n$ *(combined).* In most cases it suffices to only consider sequents of the form

$$A_1, .., A_n \vdash B \tag{2.15}$$

which means *proposition* B *is provable from propositions* $A_1, .., A_n$ *(combined).* A sequence of propositions $A_1, .., A_n$ is called a *context.* Since here we do not care about the order nor about the arity of each proposition we can consider contexts to be sets $\Gamma = \{A_1, .., A_n\}$. We use greek capital letters to denote a context.

Using the connectives new propositions ($A \wedge B$, $A \vee B$, $A \Rightarrow B$) can be build out of old ones (A, B). To get a full understanding of a newly formed proposition we need to know *how* to prove it and *what* we can prove with it. Each connective thus comes with some *inference rules* which give answers to these questions. An inference rule is written as such

$$\frac{\Gamma_1 \vdash A_1 \qquad ... \qquad \Gamma_n \vdash A_n}{\Gamma_0 \vdash A_0}$$

It means that from the sequents above the line you can deduce the sequent below the line. To know *what* we can prove from a proposition we need a rule for how the proposition can appear on the left side of a sequent, to know *how* we can prove a proposition we need a rule for how it can appear on the right side of a sequent. For the conjunction the inference rules are as follows

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\,(L\wedge) \qquad\qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(R\wedge) \tag{2.16}$$

The $(L\wedge)$-rule states, that if you can prove something under assumptions A, B, then you can also prove it under the single assumption $A \wedge B$. The $(R\wedge)$-rule states that a proof of A and a proof of B create a proof of $A \wedge B$. Both rules are very intuitive.

Instead of asking what we can prove from a proposition, we can also ask what we can do with a proof of said proposition i.e. which other proofs can we construct from them. This approach is called *natural deduction.* We call the according rules *elimination rules*, and the rules which show how to create a proof *introduction rules.* For the conjunction these are

$$\frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_i}\,(\wedge E)_{i=1,2} \qquad\qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge I) \tag{2.17}$$

The introduction rule is exactly the right rule from above. The two elimination rules state, that a proof of $A_1 \wedge A_2$ can be reduced to a proof of A_1 and a proof of A_2 respectively. Note that you can *choose* to turn a proof of $A_1 \wedge A_2$ into a proof of A_1 and you can *choose* to turn it into a proof of A_2. In contrary if given a proof of a disjunction $\Gamma \vdash A_1 \vee A_2$ you can not choose which of A_1, A_2 has been proven. This is a fundamental difference between conjunctions and disjunction which will reappear later.

Each system also comes with a set of *axioms*, these are rules, which have no sequents above the line, e.g. the axioms

$$\frac{}{A \vdash A}\,(i) \qquad\qquad \frac{}{\Gamma \vdash \top}\,(\top R)$$

which claim, that A can always be proven assuming A and that truth can be proven from anything. The following axioms are assumed in classical and intutionistic logic respectively

$$\frac{}{\neg\neg A \vdash A} \qquad\qquad \frac{}{\mathsf{0}, \Gamma \vdash A}$$

where $\neg A := A \Rightarrow \mathsf{0}$. The latter axiom is strictly weaker than the former.

The third kind of inference rules are *structural rules*, they explain the syntax of a logical system. In classical and intutionistic logic we have the *weakening* and *contraction* rules

$$\frac{A, \Gamma \vdash C}{A, B, \Gamma \vdash C}\,(w) \qquad\qquad \frac{A, A, \Gamma \vdash C}{A, \Gamma \vdash C}\,(c)$$

which state, that we can add assumptions to a proof without affecting its validity (weakening), and that duplicates in the assumptions can be eliminated (contraction).

Inference rules can be stacked to form *derivations*. We can for example stack inference rules to show that $A \wedge \top \vdash A$ and $A \vdash A \wedge \top$

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(i)}{A, \top \vdash A}\,(w)}{A \wedge \top \vdash A}\,(L\wedge) \qquad\qquad \dfrac{\dfrac{}{A \vdash A}\,(i) \quad \dfrac{}{A \vdash \top}\,(\top R)}{A \vdash A \wedge \top}\,(R\wedge)$$

We call a sequent $A \vdash B$ *valid* or *derivable* if it can be derived in above fashion. We write $A \simeq B$ if both $A \vdash B$ and $B \vdash A$ are valid. Then by above derivation we have

$$A \wedge \top \simeq A \tag{2.18}$$

making $\top$ a unit of the conjunction. Similarily we also have $A \vee \mathsf{0} \simeq A$

Linear Logic

In linear logic propositions are treated as resources. As it is with resources a proposition A can not simply be duplicated or deleted, the exact number of copies of A we're dealing with is significant. Hence in linear logic a context Γ is not a set of propositions, but rather a multiset, i.e. a set in which we keep track of the number of occurences of each element. Also the weakening and contraction rules can not hold in their current form. As a consequence of

Table 2.1: Multiplicative and additive fragment of linear logic.

	multiplicative	additive
conjunction	$\otimes$	$\&$
disjunction	$\parr$	$\oplus$
verum	1	$\top$
falsum	$\bot$	0

these changes, there are two versions of a conjunction and two versions of a disjunction, one denoting parallel availability of resources and the other non-parallel availability. Each of the four has its own distinct unit. We call the connectives standing for parallel availability and their units *multiplicative*, the others *additive*. The symbols we use for each of the connectives can be found in table 2.1. The names "multiplicative" and "additive" stem from the following distribution laws

$$X \otimes (Y \oplus Z) \simeq (X \otimes Y) \oplus (X \otimes Z)$$
$$X \parr (Y \,\&\, Z) \simeq (X \parr Y) \,\&\, (X \parr Z)$$

where again we write $X \simeq Y$ for propositions X, Y if both $X \vdash Y$ and $Y \vdash X$ are derivable.

When considering propositions as resources, this splitting of connectives into two parts is necessary. To understand why we consider the example of a vending machine. We are standing infront of a vending machine with a 2€-coin. The vending machine has multiple slots filled with various goods, these may include: white chocolate (W), dark chocolate (D), bottles of lemonade (L) and mints (M). Each slot contains multiple copies of one or more of these goods. Now the statement "with 2€ we can buy L and M" is ambiguous, it can mean two things: either the combined value of L and M is 2€ in which case we can have both available in parallel, i.e. $L \otimes M$, or each of L and M costs 2€, in which case we may have both, but not in parallel, i.e. $L \,\&\, M$. The latter case might at first glance seem like an exclusive *or* instead of an *and*, note however, that we have the choice to pick whichever of L and M we want, which is a property only a conjunction has, as we mentioned earlier in this section. The unit of parallel conjunction, 1, is an empty slot which we can tap for free, while $\top$ is an existing resource which however nobody wants, think of a pack of long expired milk. The vending machine would give it out for anything, but chosing it is never really an option, it thus acts as the unit of additive conjunction. It is a bit more difficult to explain the differences between the two disjunctions. Consider the following scenario. In one of the slots there are chocolates, they are all the same kind, but we can not detemine wether it is dark chocolate or white chocolate. We then have $D \oplus W$. If instead the slot contains both kinds of chocolate, but we do not know which it will return, we get $D \parr W$, this is the parallel disjunction[2]. The unit of additive disjunction, 0, is a theoretical resource which is impossible

[2]This explanation of the multiplicative disjunction is not perfectly accurate, as that would require a significantly more complicated example. Our example is however sufficient in providing an idea of what a disjunction denoting parallel availability of resources might look like. A more accurate example involving multiple vending machines can be found here [27].

to produce, it could thus never have been in the vending machine in the first place.

Linear logic also includes two *modalities* denoted by the symbols !, ? . Propositions of the kind $!A$ may be duplicated and contracted on the left of a sequent, propositions of the kind $?A$ on the right. The modalities are also called *exponentials*, since they convert additves into multiplicatives

$$!(X \,\&\, Y) \simeq !X \otimes !Y \qquad !\top \simeq \mathsf{1}$$
$$?(X \oplus Y) \simeq ?X \parr ?Y \qquad ?\,\mathsf{0} \simeq \bot\,.$$

With these modalities both intutionistic and classical logic may be recovered.

2.2.2 A Sequent Calculus for Linear Logic

Here we present a sequent calculus for linear logic, we exclude $\parr$, $\bot$ and ?, since we do not need either of those. This fragment of linear logic is also called *multiplicative intuitionistic linear logic*. The sequent calculus is given by the following rules.

Axioms

$$\frac{}{A \vdash A}\,(i) \qquad \frac{}{\vdash \mathsf{1}}\,(R\mathsf{1}) \qquad \frac{}{0, \Gamma \vdash C}\,(L0) \qquad \frac{}{\Gamma \vdash \top}\,(R\top)$$

Logical Rules

$$\frac{A, B, \Gamma \vdash C}{A \otimes B, \Gamma \vdash C}\,(L\otimes) \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,(R\otimes)$$

$$\frac{A_i, \Gamma \vdash C}{A_1 \,\&\, A_2, \Gamma \vdash C}\,(L\&\,)_{i=1,2} \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\,(R\&\,)$$

$$\frac{A, \Gamma \vdash C \qquad B, \Gamma \vdash C}{A \oplus B, \Gamma \vdash C}\,(L\oplus) \qquad \frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \oplus A_2}\,(R\oplus)_{i=1,2}$$

$$\frac{\Gamma \vdash A \qquad B, \Delta \vdash C}{A \multimap B, \Gamma, \Delta \vdash C}\,(L\multimap) \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\,(R\multimap)$$

$$\frac{A, \Gamma \vdash C}{!A, \Gamma \vdash C}\,(L!) \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,(R!)$$

Structural rules

$$\frac{\Gamma \vdash C}{\mathsf{1}, \Gamma \vdash C}\,(\mathsf{1}w) \qquad \frac{\Gamma \vdash C}{!A, \Gamma \vdash C}\,(!w)$$

$$\frac{!A, !A, \Gamma \vdash C}{!A, \Gamma \vdash C}\,(!c) \qquad \frac{\Gamma \vdash A \qquad A, \Delta \vdash C}{\Gamma, \Delta \vdash C}\,(cut)$$

As stated in the structural rules, contraction and weakening are only allowed for propositions of the kind $!A$ as well as for the unit 1. It can be seen, that the comma-separator on the left side of a sequent is essentially the $\otimes$-conjunction,

i.e. that the $(L\otimes)$-rule holds in reverse, too.

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(\mathrm{i}) \quad \dfrac{}{B \vdash B}\,(\mathrm{i})}{A, B \vdash A \otimes B}\,(\otimes R) \qquad A \otimes B, \Gamma \vdash C}{A, B, \Gamma \vdash C}\,(\mathrm{cut})$$

We call this new rule $(L^{-1}\otimes)$.

2.2.3 Natural Deduction for Linear Logic

The above sequent calculus is equivalent to the following system of natural deduction [19].

Axioms

$$\dfrac{}{A \vdash A}\,(i)$$

Introduction and elimination rules

$$\dfrac{}{\vdash 1}\,(1I) \qquad \dfrac{\Gamma \vdash 1 \qquad \Delta \vdash C}{\Gamma, \Delta \vdash C}\,(1E)$$

$$\dfrac{}{\Gamma \vdash \top}\,(\top I) \qquad \dfrac{\Gamma \vdash 0}{\Gamma, \Delta \vdash C}\,(0E)$$

$$\dfrac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,(\otimes I) \qquad \dfrac{\Gamma \vdash A \otimes B \qquad A, B, \Delta \vdash C}{\Gamma, \Delta \vdash C}\,(\otimes E)$$

$$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\,(\,\&\,I) \qquad \dfrac{\Gamma \vdash A_1 \,\&\, A_2 \qquad A_i, \Delta \vdash C}{\Gamma, \Delta \vdash C}\,(\,\&\,E_i)_{i=1,2}$$

$$\dfrac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \oplus A_2}\,(\oplus I_i)_{i=1,2} \qquad \dfrac{\Gamma \vdash A \oplus B \qquad A, \Delta \vdash C \qquad B, \Delta \vdash C}{\Gamma, \Delta \vdash C}\,(\oplus E)$$

$$\dfrac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\,(\multimap I) \qquad \dfrac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A \qquad B, \Theta \vdash C}{\Gamma, \Delta, \Theta \vdash C}\,(\multimap E)$$

$$\dfrac{\Gamma \vdash !A \qquad A, \Delta \vdash C}{\Gamma, \Delta \vdash C}\,(!E)$$

$$\dfrac{!A^m, \Gamma \vdash C \qquad \Delta_1 \vdash !B_1 \;\; ... \;\; \Delta_n \vdash !B_n \qquad !B_1, .., !B_n \vdash A}{\Gamma, \Delta_1, .., \Delta_n \vdash C}\,(!I)$$

where $!A^m$ denotes m occurrences of $!A$. The rules for the bang operator ! are less intuitive in this system, so we will usually be working with the !-rules from last sections sequent calculus. The $(\multimap E)$ rule can be replace by the simpler and more intuitive

$$\dfrac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}$$

Indeed the two are equivalent, the simplified version can be seen as a special case of $(\multimap E)$ where $C = B$ and Θ is empty. $(\multimap E)$ follows from above rule

by the following derivation

$$\dfrac{\dfrac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B} \qquad \dfrac{B, \Theta \vdash C}{\Theta \vdash B \multimap C}\,(\multimap I)}{\Gamma, \Delta, \Theta \vdash C}$$

By abuse of notation we call the simplified rule $(\multimap E)$, too.

2.2.4 Closed Symmetric Monoidal Theories

We claimed, that (a fragment of) linear logic describes the internal logic of quantum mechanics well. To be able to defend this claim, we need to understand what the properties of this internal logic of quantum mechanics are. For this need J.C. Baez and M. Stay introduced the notion of a *closed symmetric monoidal theory* [6], a logical analogue of a closed symmetric monoidal category. A closed symmetric monoidal theory describes the internal logic of such a category and is thus a minimal requirement for being considered an internal logic of quantum mechanics.

Definition 2.21 (Closed monoidal theory)**.** A **closed monoidal theory** consists of:

- A collection of propositions, which includes a special proposition I (the unit), as well as propositions $X \otimes Y$ and $X \multimap Y$ whenever X, Y are propositions.
- A set $X \vdash Y$ of proofs of proposition Y from proposition X for each propositions X, Y.
- 7 inference rules:

$$\dfrac{}{X \vdash X}\,(\mathrm{i}) \qquad \dfrac{X \vdash Y \qquad Y \vdash Z}{X \vdash Z}\,(\circ)$$

$$\dfrac{W \vdash X \qquad Y \vdash Z}{W \otimes Y \vdash X \otimes Z}\,(\otimes) \qquad \dfrac{W \vdash (X \otimes Y) \otimes Z}{W \vdash X \otimes (Y \otimes Z)}\,(\mathrm{a})$$

$$\dfrac{X \vdash I \otimes Y}{X \vdash Y}\,(\mathrm{l}) \qquad \dfrac{X \vdash Y \otimes I}{X \vdash Y}\,(\mathrm{r})$$

$$\dfrac{X \otimes Y \vdash Z}{Y \vdash X \multimap Z}\,(\mathrm{c})$$

- The usual equations must hold, that make the rule $(\circ)$ associative, (i) the unit of $(\circ)$, $(\otimes)$ a functor, and (a), (l), (r) and (c) natural transformations. Additionally (a), (l), (r) and $(\otimes)$ must obey the triangle and pentagon euqations.

It is clear that each of these requirements corresponds to a defining property of closed monoidal categories. A braided theory can be defined, too.

Definition 2.22 (Closed braided monoidal theory)**.**
A **closed braided monoidal theory** is a closed monoidal theory which additionally satisfies

$$\frac{W \vdash X \otimes Y}{W \vdash Y \otimes X}\,(\mathrm{b})$$

s.t. it gives rise to a natural transformation for which the hexagon equations hold.

To understand these rules better we provide a little lemma.

Lemma 2.23. *Let X, Y, W be propositions in a closed monoidal theory, X, Y being arbitrary but fixed and W arbitrary and not fixed. Then the following rules are equivalent in the sense that each can be derived from the other.*

$$\frac{W \vdash X}{W \vdash Y}\,(rule_1) \qquad\qquad \frac{}{X \vdash Y}\,(rule_2)$$

The fixing conditions encode that the X, Y from $(rule_1)$ are exactly the X, Y from $(rule_2)$ and that after fixing X, Y $(rule_1)$ holds for arbitrary W.

Proof. The derivations are simple:

$$\frac{\dfrac{}{X \vdash X}\,(i)}{X \vdash Y}\,(rule_1 \ for\ W = X) \qquad\qquad \frac{W \vdash X \qquad \dfrac{}{X \vdash Y}\,(rule_2)}{W \vdash Y}\,(\circ)$$

□

The rule (l) for example is equivalent to demanding that the judgments $I \otimes Y \vdash Y$ and $Y \vdash I \otimes Y$ are valid. Analogously for rules (r), (a) and (b).

Definition 2.24 (Closed symmetric monoidal theory)**.** A **closed symmetric monoidal theory** is a closed braided monoidal theory in which the rule (b) acts as its own inverse.

Theorem 2.25. *The $\otimes$- and $\multimap$-connectives make linear logic into a closed symmetric monoidal theory with unit $I = 1$.*

Proof. Rule (i) is satisfied and rule $(\circ)$ is a special case of the cut rule. The other rules can be derived in the following manner:

- Rules (a):

$$\frac{W \vdash (X \otimes Y) \otimes Z \qquad \dfrac{\dfrac{}{X \vdash X}\,(\mathrm{i}) \qquad \dfrac{\dfrac{}{Y \vdash Y}\,(\mathrm{i}) \qquad \dfrac{}{Z \vdash Z}\,(\mathrm{i})}{Y, Z \vdash Y \otimes Z}\,(\otimes I)}{X, Y, Z \vdash X \otimes (Y \otimes Z)}\,(\otimes I)}{X \otimes Y, Z \vdash X \otimes (Y \otimes Z)}\,(L\otimes)}{W \vdash X \otimes (Y \otimes Z)}\,(\otimes E)$$

$$\frac{W \vdash X \otimes (Y \otimes Z) \qquad \dfrac{\dfrac{}{Z \vdash Z}\,(\mathrm{i}) \qquad \dfrac{\dfrac{}{X \vdash X}\,(\mathrm{i}) \qquad \dfrac{}{Y \vdash Y}\,(\mathrm{i})}{X, Y \vdash X \otimes Y}\,(\otimes I)}{X, Y, Z \vdash (X \otimes Y) \otimes Z}\,(\otimes I)}{X, Y \otimes Z \vdash (X \otimes Y) \otimes Z}\,(L\otimes)}{W \vdash (X \otimes Y) \otimes Z}\,(\otimes E)$$

- Rule $(\otimes)$:

$$\dfrac{\dfrac{}{W \otimes Y \vdash W \otimes Y}\,(i) \qquad \dfrac{W \vdash X \qquad Y \vdash Z}{W, Y \vdash X \otimes Z}\,(\otimes I)}{W \otimes Y \vdash X \otimes Z}\,(\otimes E)$$

- Rules (l)

$$\dfrac{X \vdash 1 \otimes Y \qquad \dfrac{\dfrac{}{1 \vdash 1}\,(\mathrm{i}) \qquad \dfrac{}{Y \vdash Y}\,(\mathrm{i})}{1, Y \vdash Y}\,(1E)}{X \vdash Y}\,(\otimes E) \qquad \dfrac{\dfrac{}{\vdash 1}\,(1I) \qquad X \vdash Y}{X \vdash 1 \otimes Y}\,(\otimes I)$$

- Rules (r) can be derived analogously to rules (l), making 1 the unit of the $\otimes$-product.

- Rules (c):

$$\dfrac{Y \vdash X \multimap Z \qquad \dfrac{\dfrac{}{X \vdash X}\,(\mathrm{i}) \qquad \dfrac{}{Z \vdash Z}\,(\mathrm{i})}{X, Y \vdash Z}\,(\multimap E)}{X \otimes Y \vdash Z}\,(L\otimes)$$

$$\dfrac{\dfrac{X \otimes Y \vdash Z}{X, Y \vdash Z}\,(L^{-1}\otimes)}{Y \vdash X \multimap Z}\,(\multimap I)$$

- Rule (b):

$$\dfrac{\Gamma \vdash X \otimes Y \qquad \dfrac{\dfrac{\dfrac{}{Y \vdash Y}\,(\mathrm{i}) \qquad \dfrac{}{X \vdash X}\,(\mathrm{i})}{Y, X \vdash Y \otimes X}\,(\otimes I)}{X, Y \vdash Y \otimes X}\,(\mathrm{symm})}{\Gamma \vdash Y \otimes X}\,(\otimes E) \tag{2.19}$$

The last derivation obviously also works the other way around by simply exchanging X and Y.

According to the definition we still need to show some properties of those rules. We demanded for example that $(\circ)$ is associative which means that the following two derivations result in equal judgments

$$\dfrac{W \vdash X \qquad \dfrac{X \vdash Y \qquad Y \vdash Z}{X \vdash Z}\,(\circ)}{W \vdash Z}\,(\circ) \qquad \dfrac{\dfrac{W \vdash X \qquad X \vdash Y}{W \vdash Y}\,(\circ) \qquad Y \vdash Z}{W \vdash Z}\,(\circ)$$

We have however not introduced any way of distinguishing between two judgments $\pi : W \vdash Z$, $\sigma : W \vdash Z$. Since in our usage of judgments we only cared about what propositions its left and right side contained, we only really have the option of calling judgments equal whenever their left and right sides coincide respectively. Thus $\pi = \sigma$ and similarily all other properties left to prove hold trivially as well. Note however that a theory build upon linear logic, particularily the typing system we introduce in the next chapter, might indeed provide ways of distinguishing between judgments like π and σ, in which case their equality has to be explicitely proven. □

2.2.5 Further properties of Linear Logic

Next we discuss some properties of linear logic, that will become relevant in the following chapters.

In the multiplicative fragment there is no duplication and no deletion, i.e.

$$X \nvdash X \otimes X \qquad\qquad X \nvdash 1\,,$$

in the additive part however the two analogous sequents

$$X \vdash X \,\&\, X \qquad\qquad X \vdash \top$$

are both derivable. The bang operator ! makes duplication and deletion possible in the multiplicative part, i.e. the following two are valid

$$!X \vdash !X \otimes !X \qquad\qquad !X \vdash 1\,. \tag{2.20}$$

The derivations are as follows:

$$\dfrac{\dfrac{\dfrac{}{!X \vdash !X}\,(i) \quad \dfrac{}{!X \vdash !X}\,(i)}{!X, !X \vdash !X \otimes !X}\,(\otimes I)}{!X \vdash !X \otimes !X}\,(!c) \qquad\qquad \dfrac{\dfrac{}{\vdash 1}\,(1I)}{!X \vdash 1}\,(!w) \tag{2.21}$$

On propositions of the kind $!A$ both conjunctions coincide i.e.

$$\dfrac{!\Gamma \vdash !X \,\&\, !Y}{!\Gamma \vdash !X \otimes !Y}$$

Note that it is crucial here that all propositions are of the kind $!A$, not only X, Y.
We showed above, that deletion of propositions of the kind $!X$ is possible, we give the more generalized rule the name $\nabla^!$:

$$\dfrac{}{!\Gamma \vdash 1}\,(\nabla^!) \tag{2.22}$$

then we can prove a very important lemma.

Lemma 2.26. *The following equivalence holds*

$$(!w) \wedge (1I) \qquad \Longleftrightarrow \qquad (\nabla^!)\,. \tag{2.23}$$

That means that if the rest of the rules stay the same we can replace $(!w)$ *and* $(1I)$ *by* $(\nabla^!)$*.*

Proof. "⇒": Let n be the number of propositions in Γ, then

$$\dfrac{\dfrac{}{\vdash 1}\,(1I)}{!\Gamma \vdash 1}\,(!w)\text{ n times}$$

is the derivation we seek.
"⇐": $(1I)$ is just a special case of $(\nabla^!)$ for an empty context Γ. The derivation of $(!w)$ from $(\nabla^!)$ would be too wide to fit on this paper, we thus divide it into

two steps. We first derive

$$\frac{\Gamma \vdash C}{!A, \Gamma \vdash C \otimes 1} \qquad \text{and then} \qquad \frac{!A, \Gamma \vdash C \otimes 1}{!A, \Gamma \vdash C}$$

so that put together they create a derivation of $(!w)$. The first derivation goes as follows

$$\dfrac{\dfrac{\Gamma \vdash C \qquad \dfrac{}{!A \vdash !A}\,(i)}{!A, \Gamma \vdash C \otimes !A}\,(\otimes I) \qquad \dfrac{\dfrac{}{C \vdash C}\,(i) \qquad \dfrac{}{!A \vdash 1}\,(\nabla^!)}{C \otimes !A \vdash C \otimes 1}\,(\otimes)}{!A, \Gamma \vdash C \otimes 1}\,(cut)$$

the second derivation is:

$$\dfrac{!A, \Gamma \vdash C \otimes 1 \qquad \dfrac{\dfrac{}{C \otimes 1 \vdash C \otimes 1}\,(i) \qquad \dfrac{\dfrac{}{1 \vdash 1}\,(i) \qquad \dfrac{}{C \vdash C}\,(i)}{C, 1 \vdash C}\,(1E)}{C \otimes 1 \vdash C}\,(\otimes E)}{!A, \Gamma \vdash C}\,(cut)$$

This concludes the proof. □

This lemma allows us to choose freely wether we want to define our logic with rules $(!w)$ and $(1I)$ or rather only with rule $(\nabla^!)$.

The linear implication is contravariant in the first argument and covariant in the second argument, meaning that from $A \vdash B$ we can derive $C \multimap A \vdash C \multimap B$ as well as $B \multimap C \vdash A \multimap C$.

$$\dfrac{\dfrac{\dfrac{}{C \vdash C}\,(i) \qquad A \vdash B}{C, C \multimap A \vdash B}\,(L\multimap)}{C \multimap A \vdash C \multimap B}\,(\multimap I) \qquad \dfrac{\dfrac{A \vdash B \qquad \dfrac{}{C \vdash C}\,(i)}{A, B \multimap C \vdash C}\,(L\multimap)}{B \multimap C \vdash A \multimap C}\,(\multimap I) \tag{2.24}$$

Let's take a look at how the bang operator "!" interacts with the multiplicative product. In particular we want to compare the propositons $!A \otimes !B$ and $!(A \otimes B)$. We have seen, that ! makes duplication possible. In the case of $!A \otimes !B$ we thus have a pair of duplicable propositions. In the case of $!(A \otimes B)$ we have a duplicable pair of propositions, each of which might however not be duplicable. That means in the former case we have for example derivable sequents of the form $!A \otimes !B \vdash !A \otimes !A \otimes !B$, which duplicate only a single proposition, something that is not possible from $!(A \otimes B)$. Additionally the sequent $!A \otimes !B \vdash !(A \otimes B)$ is derivable.

$$\dfrac{\dfrac{}{!A \otimes !B \vdash !A \otimes !B}\,(i) \qquad \dfrac{\dfrac{\dfrac{}{A \vdash A}\,(i) \qquad \dfrac{\dfrac{}{B \vdash B}\,(i)}{!B \vdash B}\,(!L)}{!A, !B \vdash A \otimes B}\,(\otimes I)}{!A, !B \vdash !(A \otimes B)}\,(R!)}{!A \otimes !B \vdash !(A \otimes B)}\,(\otimes E) \tag{2.25}$$

Lastly there is a bijection we want to shed light on. Every sequent of the form $\Gamma \vdash A$ can be identified with a sequent of the form $\Gamma \vdash 1 \multimap A$. This can

be derived the following way:

$$\dfrac{\dfrac{\Gamma \vdash A}{1, \Gamma \vdash A}\,(1L)}{\Gamma \vdash 1 \multimap A}\,(\multimap I) \tag{2.26}$$

and

$$\dfrac{\Gamma \vdash 1 \multimap A \qquad \dfrac{}{\vdash 1}\,(1E) \qquad \dfrac{}{A \vdash A}\,(i)}{\Gamma \vdash A}\,(\multimap E) \tag{2.27}$$

Since this identification will become relevant in the next chapter, we show some of its properties. The bijection preserves validity, i.e. if $A \vdash B$, then $1 \multimap A \vdash 1 \multimap B$. The derivation is simple:

$$\dfrac{\dfrac{\dfrac{}{1 \vdash 1}\,(i) \qquad A \vdash B}{1, 1 \multimap A \vdash B}\,(L\multimap)}{1 \multimap A \vdash 1 \multimap B}\,(\multimap I) \tag{2.28}$$

Furthermore we have $A \otimes (1 \multimap B) \vdash 1 \multimap (A \otimes B)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(i) \qquad \dfrac{\dfrac{}{\vdash 1}\,(1I) \qquad \dfrac{}{B \vdash B}\,(i)}{1 \multimap B \vdash B}\,(L\multimap)}{A \otimes (1 \multimap B) \vdash A \otimes B}\,(\otimes I)}{1, A \otimes (1 \multimap B) \vdash A \otimes B}\,(1w)}{A \otimes (1 \multimap B) \vdash 1 \multimap (A \otimes B)}\,(\multimap I) \tag{2.29}$$

Next we show, that the rule

$$\dfrac{A \vdash B \multimap C}{A \otimes B \vdash 1 \multimap C}$$

holds. The derivation goes as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{A \vdash B \multimap C \qquad \dfrac{}{B \vdash B}\,(i) \qquad \dfrac{}{C \vdash C}\,(i)}{A, B \vdash C}\,(\multimap E)}{A \otimes B \vdash C}\,(L\otimes)}{1, A \otimes B \vdash C}\,(1w)}{A \otimes B \vdash 1 \multimap C}\,(\multimap I) \tag{2.30}$$

That concludes our discourse on the properties of linear logic.

The attentive reader will have also noted the non-properties of linear logic. What is missing, in comparison to our discussion in the previous section, is a †-functor and a notion of a strong dual. The former is mostly incompatible with logic. Indeed the dagger would have to satisfy $(A \vdash B)^\dagger = B \vdash A$, however the validity of $B \vdash A$ can never be derived from $A \vdash B$. Anyway the dagger will not be necessary in the following chapters. A candidate for a dual proposition is $A^* := A \multimap 1$. But that is only a weak dual, since there is no co-evaluation i.e. in general $1 \nvdash A^* \otimes A$. In quantum mechanics the dual of a particle is its anti-particle. Since however anti-matter is not used in quantum computers, a notion of a dual will not be necessary in the following chapters either. The construction of a logic that does include strong duals is sketched in section 4.1.1.

Chapter 3

Quantum Lambda Calculus

In the previous chapter we have presented how linear logic is capable of describing quantum mechanics internally. It is thus obvious, that a lambda calculus for quantum computation needs to be founded on linear logic. In the first section of this chapter we introduce a typed lambda calculus whose types largely follow a linear logic with some variations. In the second section we consider the semantics of this language categorically and compare them to the categories from the previous chapter.

3.1 Construction of a Quantum Lambda Calculus

Here we construct a quantum lambda calculus. It is the calculus developed by B. Valiron [25] extended by coproducts.

3.1.1 Terms

The terms in the quantum lambda calculus are as follows.

Definition 3.1 (Terms)**.**

$$\begin{aligned} Term \;\; M,N,P \;\; ::= \;\; & c \mid x \mid MN \mid \lambda x.M \mid \langle M,N\rangle \mid \star \mid \\ & let \star = M \; in \; N \mid let \; \langle x,y\rangle = M \; in \; N \mid inj_l(M) \mid \\ & inj_r(M) \mid match \; P \; with \; (x \mapsto M \mid y \mapsto N) \end{aligned}$$

Here x ranges over an infinite set of variables. The term $\lambda x.M$ stands for the function $x \mapsto M$, while MN denotes application of M to argument N. $\langle M,N\rangle$ is a tuple consisting of M and N, $\star$ the 0-tuple. In $let \; \langle x,y\rangle = M \; in \; N$ the term M returns a pair $\langle V,W\rangle$, then in N variables x and y are replaced by V and W respectively. $inj_l(M)$ and $inj_r(M)$ are the left and right inclusion functions into a disjoint union, while the term $match \; P \; with \; (x \mapsto M \mid y \mapsto N)$ makes a case distinction proceeding with M if P is of the form $inj_l(x)$ and with N if it is of the form $inj_r(y)$. We do in that term always assume that $x \neq y$. Finally c ranges over a set of term constants, which might in principal be arbitrary. For quantum computation the following are of use:

- *new* is a term that prepares a quantum state. It takes as input a classical bit which might be $\mathfrak{o}$ or $\mathfrak{1}$ (see Notation 3.4 below) and accordingly outputs a qubit in state $|0\rangle$ or $|1\rangle$.
- a collection of unitary gates U each manipulating one or multiple qubits.
- a function *meas* representing the measurement, which takes as input a qubit, measures it in standard basis $\{|0\rangle, |1\rangle\}$ and returns the measured value as a (classical) bit.

Definition 3.2 (Free Variables, Bound Variables)**.** We denote the set of **free variables** of a Term M by $FV(M)$. It is defined inductively as follows

$$\begin{aligned} FV(c) := FV(\star) &:= \emptyset \\ FV(x) &:= \{x\} \\ FV(MN) := FV(\langle M, N\rangle) &:= FV(M) \cup FV(N) \\ FV(let\ \star = M\ in\ N) &:= FV(M) \cup FV(N) \\ FV(\lambda x.M) &:= FV(M)\backslash\{x\} \\ FV(let\ \langle x, y\rangle := M\ in\ N) &:= FV(M) \cup (FV(N)\backslash\{x, y\}) \\ FV(inj_l(M)) := FV(inj_r(M)) &:= FV(M) \end{aligned}$$

$$\begin{aligned} &FV\left(match\ P\ with\ \ (x \mapsto M \mid y \mapsto N)\right) := \\ &\qquad FV(P) \cup (FV(M)\backslash\{x\}) \cup (FV(N)\backslash\{y\})\,. \end{aligned} \tag{3.1}$$

The non-free variables of a term are called **bound**.

Following common practice we consider terms to be equal if they're identical up to renaming bound variables e.g. $\lambda x.f(x) = \lambda y.f(y)$ for any suitable f. This is known as α-equivalence.

Definition 3.3 (Substitution)**.** Let x be a variable and P a term. We define substitution of x by P inductively. The base cases are

$$\begin{aligned} x[P/x] &:= P \\ y[P/x] &:= y \quad (if\ y \neq x) \\ c[P/x] &:= c \\ \star[P/x] &:= \star \end{aligned}$$

Now assume, that substitution is defined for terms M, N, N_1, N_2, then we define

$$\begin{aligned} (MN)[P/x] &:= (M[P/x])(N[P/x]) \\ (\langle M, N\rangle)[P/x] &:= \langle M[P/x], N[P/x]\rangle \\ (inj_l(M))[P/x] &:= inj_l(M[P/x]) \\ (inj_r(M))[P/x] &:= inj_r(M[P/x]) \\ (\lambda x.M)[P/x] &:= \lambda x.M \\ (\lambda y.M)[P/x] &:= \lambda y.(M[P/x]) \quad (if\ y \neq x\ and\ y \notin FV(P)) \\ (let\ \star = M\ in\ N)[P/x] &:= let\ \star = M[P/x]\ in\ N[P/x] \end{aligned}$$

Two cases remain. Substitution in the term $match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2)$ is defined as follows. If $x = y_1$ and $y_2 \notin FV(P)$

$$(match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2))[P/x] := \\ match\ M[P/x]\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2[P/x])$$

if $x = y_2$ and $y_1 \notin FV(P)$

$$(match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2))[P/x] := \\ match\ M[P/x]\ with\ (y_1 \mapsto N_1[P/x] \mid y_2 \mapsto N_2)$$

and if $y_1 \neq x \neq y_2$

$$(match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2))[P/x] := \\ match\ M[P/x]\ with\ (y_1 \mapsto N_1[P/x] \mid y_2 \mapsto N_2[P/x])\ .$$

For the term $let\ \langle y, z\rangle = M\ in\ N$ substitution works the following way. If $x \in \{y, z\}$

$$(let\ \langle y, z\rangle = M\ in\ N)[P/x]\ :=\ let\ \langle y, z\rangle = M[P/x]\ in\ N$$

and if $x \notin \{y, z\}$ and $y, z \notin FV(P)$

$$(let\ \langle y, z\rangle = M\ in\ N)[P/x]\ :=\ let\ \langle y, z\rangle = M[P/x]\ in\ N[P/x]\,.$$

Additionally we simplify the notation using the following conventions:

Notation 3.4.

$$\begin{aligned}
\lambda x_1 x_2..x_n.M &:= \lambda x_1.\lambda x_2...\lambda x_n.M \\
M_1 M_2 M_3..M_n &:= (..((M_1 M_2)M_2)..M_n) \\
\langle M_1, M_2, .., M_{n-1}, M_n\rangle &:= \langle M_1, \langle M_2, ..\langle M_{n-1}, M_n\rangle..\rangle\rangle \\
\lambda\langle x, y\rangle.M &:= \lambda z.(let\ \langle x, y\rangle = z\ in\ M) \\
\lambda\star.M &:= \lambda z.(let\ \star = z\ in\ M)\ \ \text{(where z is fresh)} \\
let\ x = M\ in\ N &:= (\lambda x.N)M \\
\mathfrak{o} &:= inj_r(\star) \\
\mathfrak{1} &:= inj_l(\star) \\
if\ P\ then\ M\ else\ N &:= match\ P\ with\ (x \mapsto M \mid y \mapsto N)\ \ \text{(with } x, y \text{ fresh)}
\end{aligned}$$

Lemma 3.5. *Let M, P be terms and $x \notin FV(M)$. Then $M[P/x] = M$.*

Proof. The proof is done by structural induction on M. It is non-constructive. We begin with the base cases.

$\underline{Case\ M \equiv y}$: Since $x \notin FV(M)$ we necessarily have $x \neq y$. Thus $M[P/x] = y[P/x] = y = M$.

$\underline{Case\ M \equiv c}$: $M[P/x] = c[P/x] = c = M$.

Case $M \equiv c$: $M[P/x] = \star[P/x] = \star = M$.

Now assume the lemma holds for terms N, N_0, N_1, N_2.

Case $M \equiv inj_l(N)$: $M[P/x] = inj_l(N)[P/x] = inj_l(N[P/x])$. Since $x \notin FV(M) = FV(N)$ we may conclude by the induction hypothesis that $N[P/x] = N$ and thus $M[P/x] = inj_l(N) = M$.

Case $M \equiv inj_r(N)$: Analogously.

Case $M \equiv \lambda y.N$: If $y = x$ we have $M[P/x] = (\lambda x.N)[P/x] = \lambda x.N = M$. If $y \neq x$ we have $M[P/x] = (\lambda y.N)[P/x] = \lambda x.(N[P/x])$. Since $x \notin FV(M) = FV(N)\backslash\{x\}$ and $x \neq y$ we must have $x \notin FV(N)$, so that we may conclude by the induction hypothesis that $N[P/x] = N$ and thus $M[P/x] = \lambda x.N = M$.

Case $M \equiv N_1 N_2$: $M[P/x] = (N_1 N_2)[P/x] = (N_1[P/x])(N_2[P/x])$. Since $x \notin FV(M) = FV(N_1) \cup FV(N_2)$ we have $x \notin FV(N_i)$, $i = 1{,}2$, and we may conclude by the induction hypothesis that $N_i[P/x] = N_i$, $i = 1{,}2$, and thus $M[P/x] = N_1 N_2 = M$.

Case $M \equiv \langle N_1, N_2 \rangle$: $M[P/x] = \langle N_1, N_2 \rangle[P/x] = \langle N_1[P/x], N_2[P/x] \rangle$. Since $x \notin FV(M) = FV(N_1) \cup FV(N_2)$ we have $x \notin FV(N_i)$, $i = 1{,}2$, and we may conclude by the induction hypothesis that $N_i[P/x] = N_i$, $i = 1{,}2$, and thus $M[P/x] = \langle N_1, N_2 \rangle = M$.

Case $M \equiv$ *let* $\star = N_1$ *in* N_2 : $M[P/x] = (let \star = N_1 \; in \; N_2)[P/x] = let \star = N_1[P/x] \; in \; N_2[P/x]$. Since $x \notin FV(M) = FV(N_1) \cup FV(N_2)$ we have $x \notin FV(N_i)$, $i = 1{,}2$, and we may conclude by the induction hypothesis that $N_i[P/x] = N_i$, $i = 1{,}2$, and thus $M[P/x] = let \star = N_1 \; in \; N_2 = M$.

Case $M \equiv$ *match* N_0 *with* $(y_1 \mapsto N_1 \mid y_2 \mapsto N_2)$: It is $FV(M) = FV(N_0) \cup (FV(N_1)\backslash\{y_1\}) \cup (FV(N_2)\backslash\{y_2\})$ and we thus have $x \notin FV(N_0)$ and $x \notin FV(N_i)\backslash\{y_i\}$, $i = 1{,}2$. By the induction hypothesis we can conclude, that $N_0[P/x] = N_0$. We now have three subcases. If $x = y_1$ we have $x \notin FV(N_2)$, since in particular $x \neq y_2$, and thus by the induction hypothesis $N_2[P/x] = N_2$. Then $M[P/x] = match \; N_0[P/x] \; with \; (y_1 \mapsto N_1 \mid y_2 \mapsto N_2[P/x]) = M$. If instead $x = y_2$ proceed analogously. If however $y_1 \neq x \neq y_2$ we have $x \notin FV(N_i)$ for both $i = 1{,}2$. Thus by the induction hypothesis $N_i[P/x] = N_i$, $i = 1{,}2$. We then have that $M[P/x] = match \; N_0[P/x] \; with \; (y_1 \mapsto N_1[P/x] \mid y_2 \mapsto N_2[P/x] = M$.

Case $M \equiv$ *let* $\langle y, z \rangle = N_1$ *in* N_2 : Since $x \notin FV(M)$ and $FV(M) = FV(N_1) \cup (FV(N_2)\backslash\{y, z\})$ we can conclude that $x \notin FV(N_1)$. Thus by induction hypothesis $N_1[P/x] = N_1$. If $x \in \{y, z\}$ we then have $M[P/x] = (let \; \langle y, z \rangle = N_1 \; in \; N_2)[P/x] = let \; \langle y, z \rangle = N_1[P/x] \; in \; N_2 = let \; \langle y, z \rangle = N_1 \; in \; N_2 = M$. If $x \notin \{y, z\}$ then also $x \notin FV(N_2)$ and by induction hypothesis $N_2[P/x] = N_2$. We then have $M[P/x] = (let \; \langle y, z \rangle = N_1 \; in \; N_2)[P/x] = let \; \langle y, z \rangle = N_1[P/x] \; in \; N_2[P/x] = let \; \langle y, z \rangle = N_1 \; in \; N_2 = M$. If $x \notin \{y, z\}$ then also $x \notin FV(N_2)$

That concludes the proof. □

3.1.2 Types and Subtyping

Of course not all terms that can be produced in this fashion are admissible. Some are not allowed for physical reasons, e.g. $\lambda x.\langle x, x\rangle$ which duplicates x might be problematic whenever x stands for a quantum state, others don't make sense to begin with, the term MN for example only makes sense when M is a function and N the right type of input. Thus to ensure admissibility of terms a type system needs to be introduced. To account in particular for non-duplicability it is based on linear logic.

Definition 3.6 (Types)**.**

$$\mathit{Type}\quad A, B \quad ::= \quad \mathit{qbit} \mid !A \mid A \multimap B \mid 1 \mid A \otimes B \mid A \oplus B$$

The type *qbit* represents the one-qubit states. By $!A$ we denote the *subtype* of duplicaple/reusable values of type A. For types A and B, $A \multimap B$ is the type of functions from A to B, $A \otimes B$ the type of pairs of a value of type A and a value of type B and $A \oplus B$ the disjoint union of A and B. 1 is a singleton. We introduce some shorthand notations.

Notation 3.7.

$$\begin{aligned}
!^n A &:= !!..!!A \quad \text{for } n \text{ repetitions of } !\\
A^{\otimes n} &:= (..(A \otimes A).. \otimes A) \quad \text{for the } n\text{-fold tensor product}\\
A^{\oplus n} &:= (..(A \oplus A).. \oplus A) \quad \text{analogously}\\
\mathit{bit} &:= 1 \oplus 1
\end{aligned}$$

Translating the descriptions of our constants *new*, U, *meas* we find that they're supposed to be of the types:

$$\mathit{new} : \mathit{bit} \multimap \mathit{qbit} \qquad U : \mathit{qbit}^{\otimes n} \multimap \mathit{qbit}^{\otimes n} \qquad \mathit{meas} : \mathit{qbit} \multimap !\mathit{bit}\,, \tag{3.2}$$

where n is the arity of the respective gate. It was already mentioned, that $!A$ is supposed to be a subtype of A in the sense that every term of type $!A$ in particular also has type A. The according subtyping relation, which we denote by $\leq:$, can be defined by a set of rules.

Definition 3.8 (Subtyping Relation)**.**
The **subtyping relation** $\leq:$ is defined as the smallest relation on types, that satisfies the rules in table 3.1. We also define the relation $\doteqdot$ on types to mean $A \doteqdot B \;\equiv\; (A \leq: B) \wedge (B \leq: A)$.

To eradicate confusion about the condition $(m = 0) \vee (n \geq 1)$ (cmp. table 3.1) let's analyze for a type A the expression $!^n A \leq: !^m A$. In the case $m = 0$ it becomes $!^n A \leq: A$ as wished. In the case $n \geq 1$ we obtain $!^n A \leq: !^m A$ for all $m \in \mathbb{N}$, in particular $!^n A \doteqdot !A$ for all $n \in \mathbb{N}$ which amounts to saying that we define reusability to mean *infinitely reusable* as opposed to *reusable once*. This is the whole secret behind that condition.

Lemma 3.9. *The subtyping relation* $\leq:$ *is reflexive and transitive.* □

This also implies that $\doteqdot$ is an equivalence relation. The following two lemmata will be of use.

Table 3.1: Inference rules of the subtyping relation. Here $m, n \in \mathbb{N}$ and we always assume $(m = 0) \vee (n \geq 1)$.

$$\frac{}{!^n qbit \leq: !^m qbit}\ (qbit) \qquad \frac{}{!^n 1 \leq: !^m 1}\ (1)$$

$$\frac{A_1 \leq: B_1 \qquad A_2 \leq: B_2}{!^n(A_1 \otimes A_2) \leq: !^m(B_1 \otimes B_2)}\ (\otimes) \qquad \frac{A_1 \leq: B_1 \qquad A_2 \leq: B_2}{!^n(A_1 \oplus A_2) \leq: !^m(B_1 \oplus B_2)}\ (\oplus)$$

$$\frac{A \leq: A' \qquad B \leq: B'}{!^n(A' \multimap B) \leq: !^m(A \multimap B')}\ (\multimap)$$

Lemma 3.10. *Let A, B be types such that $A \leq: !B$, then there is a type A' such that $A = !A'$. Consequently, if $A \leq: B$ and A is not of the form $!A'$ for some A', then B is not of the form $!B'$ for some B' either.*

Proof. Follows immediately from the cases $m \neq 0$ and $n = 0$ in table 3.1. □

Lemma 3.11. *Let A, B be types. If $!B \leq: A$ then $!B \leq: !A$.*

Proof. If there is a type A' such that $A = !A'$ the proof becomes trivial:

$$!B \leq: A = !A' \div !!A' = !A\,.$$

So assume that A is not of the form $!A'$. We proceed by structural induction on the type A, beginning with the base cases.

Case $A \equiv qbit$: The only applicable subtyping rule is $(qbit)$. We must thus have $!B = !^{n+1} qbit$, $n \in \mathbb{N}$, and thus by rule $(qbit)$ $!B = !^{n+1} qbit \leq: !qbit = !A$.

Case $A \equiv 1$: The only applicable subtyping rule is (1). We must thus have $!B = !^{n+1}1$, $n \in \mathbb{N}$, and thus by rule (1) $!B = !^{n+1}1 \leq: !1 = !A$.

Assuming the lemma holds for types A_1, A_2, we continue with the induction steps.

Case $A \equiv A_1 \otimes A_2$: The only applicable subtyping rule is $(\otimes)$. There must thus be types B_1, B_2 with $B_i \leq: A_i$, $i = 1{,}2$ such that $!B = !^{n+1}(B_1 \otimes B_2)$, $n \in \mathbb{N}$. Thus by rule $(\otimes)$ $!B = !^{n+1}(B_1 \otimes B_2) \leq: !(A_1 \otimes A_2) = !A$

Case $A \equiv A_1 \oplus A_2$: Analogously.

Case $A \equiv A_1 \multimap A_2$: The only applicable subtyping rule is $(\multimap)$. There must thus be types B_1, B_2 with $A_1 \leq: B_2$ and $B_2 \leq: A_2$ such that $!B = !^{n+1}(B_1 \multimap B_2)$, $n \in \mathbb{N}$. Thus by rule $(\otimes)$ $!B = !^{n+1}(B_1 \multimap B_2) \leq: !(A_1 \multimap A_2) = !A$

By induction the lemma holds. The proof is non-constructive. □

Table 3.2: Typing rules. Here A_c denotes the type of constant c (cmp. (3.2)).

$$\frac{A \leq: B}{!\Delta, x : A \vdash x : B}\ (ax_1) \qquad \frac{!A_c \leq: B}{!\Delta \vdash c : B}\ (ax_2)$$

$$\frac{\Gamma \vdash M : !^n A}{\Gamma \vdash inj_l(M) : !^n(A \oplus B)}\ (\oplus I_1) \qquad \frac{\Gamma \vdash M : !^n B}{\Gamma \vdash inj_r(M) : !^n(A \oplus B)}\ (\oplus I_2)$$

$$\frac{!\Delta, \Theta \vdash P : !^n(A \oplus B) \qquad !\Delta, \Gamma, x : !^n A \vdash M : C \qquad !\Delta, \Gamma, y : !^n B \vdash N : C}{!\Delta, \Theta, \Gamma \vdash match\ P\ with\ (x \mapsto M \mid y \mapsto N) : C}\ (\oplus E)$$

$$\frac{!\Delta, \Gamma_1 \vdash M : A \multimap B \qquad !\Delta, \Gamma_2 \vdash N : A}{!\Delta, \Gamma_1, \Gamma_2 \vdash MN : B}\ (\multimap E)$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ (\multimap I_1) \qquad \frac{!\Delta, x : A \vdash M : B}{!\Delta \vdash \lambda x.M : !^{n+1}(A \multimap B)}\ (\multimap I_2)$$

$$\frac{!\Delta, \Gamma_1 \vdash M_1 : !^n A_1 \qquad !\Delta, \Gamma_2 \vdash M_2 : !^n A_2}{!\Delta, \Gamma_1, \Gamma_2 \vdash \langle M_1, M_2 \rangle : !^n(A_1 \otimes A_2)}\ (\otimes I)$$

$$\frac{!\Delta, \Gamma_1 \vdash M : !^n(A_1 \otimes A_2) \qquad !\Delta, \Gamma_2, x_1 : !^n A_1, x_2 : !^n A_2 \vdash N : A}{!\Delta, \Gamma_1, \Gamma_2 \vdash let\ \langle x_1, x_2 \rangle = M\ in\ N : A}\ (\otimes E)$$

$$\frac{!\Delta, \Gamma \vdash N : 1 \qquad !\Delta, \Theta \vdash M : A}{!\Delta, \Gamma, \Theta \vdash\ let \star = N\ in\ M : A}\ (1E) \qquad \frac{}{!\Delta \vdash \star : !^n 1}\ (\nabla^!)$$

3.1.3 Typing Rules

In this subsection we describe a formalism with which we can decide the type of a term.

Definition 3.12 (Typing Context)**.** A **typing context** Γ is a finite set $\{x_1 : A_1, .., x_n : A_n\}$ of ordered pairs of a variable and a type, which we denoted by $x_i : A_i$, such that no variable is repeated twice. We may write $|\Gamma|$ for the set of variables $|\Gamma| = \{x_1, .., x_n\}$ and $\Gamma(x_i)$ for the type of variable x_i, $\Gamma(x_i) = A_i$. Additionally we write $!\Gamma$ in cases where all variables are of re-usable type and Γ, Γ' for the union of two disjoint typing contexts.

Definition 3.13 (Typing judgement)**.** For a typing context Γ, a term M and a type A, a **typing judgement** is an expression of the form $\Gamma \vdash M : A$. We call a typing judgement **valid** if it follows from the rules in table 3.2.

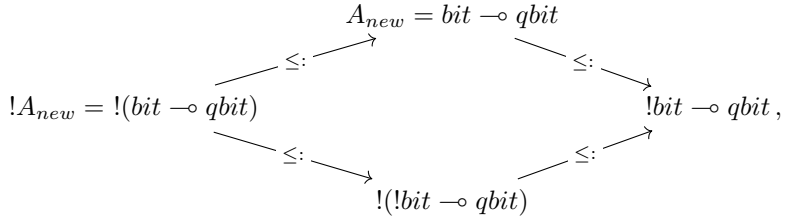
Having formally introduced the typing rules, we can now also see that

$$\vdash \mathfrak{o} : !^n bit \qquad\qquad \vdash \mathfrak{1} : !^n bit \tag{3.3}$$

are valid.

Let's devote a moment to explaining why a variable may not repeat twice in a context. Allowing multiple appearances of a variable would allow for judgements of the form $x : A, x : A \vdash \langle x, x\rangle : A \otimes A$, which upon applying rule $(\multimap I_1)$ can be turned into $x : A \vdash \lambda x.\langle x, x\rangle : A \otimes A$. This expression is obviously problematic, as it behaves a lot like a duplicating function, which we're trying to avoid at all costs. We might e.g. apply rule $(\multimap E)$ to it and some valid $\Gamma \vdash V : A$ and obtain $\Gamma, x : A \vdash (\lambda x.\langle x, x\rangle)V : A \otimes A$. Now it is unclear how $(\lambda x.\langle x, x\rangle)V$ can mean something that does not duplicate V. The condition of non repeated variables takes care of these issues. Alternative solutions exist, but they make the language too complicated to be considered a serious option.

The rule (ax_2) in table 3.2 was set up such that $!A_c$ is the most specific type of the respective constant c, meaning that if $!A_c \not\leq: B$, then $\nvdash c : B$. It should however be noted that there are types other then A_c of which $!A_c$ is a subtype of. Examining e.g. the very important example of *new* we have that



where the arrows denote subtyping. This diagram is exhaustive, meaning that if there is any type B such that $!A_{new} \leq: B$, then B is equal $(\div)$ to one of the types in the diagram. This is an immediate consequence of the subtyping rules $(qbit)$ and $(\multimap)$ (cmp. table 3.1). In particular this example shows, that no reusable qubit can be created.

We've put a lot of work into controlling duplication. So the following lemma is due.

Lemma 3.14. *Assume that the judgement $x : A \vdash \langle x, x\rangle : A \otimes A$ is valid, then there is a type A' such that $A = !A'$.*

Proof. The judgement can only have been introduced by rule $(\otimes I)$. Thus we must have valid judgements $!\Delta, \Gamma_1 \vdash x : A$, $!\Delta, \Gamma_2 \vdash x : A$ such that $(!\Delta, \Gamma_1, \Gamma_2) = (x : A)$. We have to make a case destinction.

<u>*Case 1*</u> : $x : A$ *is in* $!\Delta$. Per definition all types in $!\Delta$ are reusable, so there is a type A' such that $A = !A'$.

<u>*Case 2*</u> : $x : A$ *is in* Γ_1. That implies that $!\Delta$ and Γ_2 are empty. So the judgements are of the form $x : A \vdash x : A$ and $\vdash x : A$. However the latter judgement can not be valid as a typing $x : A$ can only be introduced via (ax_1), which does not allow an emtpy context. This case can thus never occur.

<u>*Case 3*</u> : $x : A$ *is in* Γ_2. Analogously to above this case can never occur.

That concludes the proof. It is non-constructive. □

Corollary 3.15. $x : qbit \nvdash \langle x, x\rangle : qbit \otimes qbit$

Proof. The type *qbit* is not reusable. □

This type system is based on linear logic, however with the choice to handle the !-operator differently than we did in the previous chapter, we introduced some notable differences. To ensure that the derivations we made in the linear logic are reproducable here, we need to show that the rules applied still hold. This includes the rules (i), $(!L)$ and $(!c)$. We also derive the cut-rule, to show what it looks like in the term-language.

Lemma 3.16. *The following rules are derivable in our typing system*

$$\frac{}{x : A \vdash x : A}\,(i) \qquad \frac{\Theta \vdash N : A \qquad \Gamma, x : A \vdash M : B}{!\Delta, \Theta, \Gamma \vdash \ let\ x = N\ in\ M : B}\,(cut)$$

$$\frac{x : A, \Gamma \vdash M : B}{x : !A, \Gamma \vdash M : B}\,(!L) \qquad \frac{x : !A, y : !A, \Gamma \vdash M : B}{y : !A, \Gamma \vdash \ let\ x = y\ in\ M : B}\,(!c)$$

Proof. The derivations are as follows

$$\frac{\dfrac{\overline{A \leq: A}}{x : A \vdash x : A}\,(ax_1) \qquad \dfrac{\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\,(\multimap I_1) \qquad \Theta \vdash N : A}{\Theta, \Gamma \vdash \ let\ x = N\ in\ M : B}\,(\multimap E)$$

$$\frac{\dfrac{\dfrac{\overline{!A \leq: A}}{x : !A \vdash x : A}\,(ax_1) \qquad x : A, \Gamma \vdash M : B}{x : !A, \Gamma \vdash M : B}\,(cut)$$

$$\frac{\dfrac{\dfrac{x : !A, y : !A, \Gamma \vdash M : B}{y : !A, \Gamma \vdash \lambda x.M : !A \multimap C}\,(\multimap I_1) \qquad \dfrac{}{y : !A \vdash y : !A}\,(i)}{y : !A, \Gamma \vdash \ let\ x = y\ in\ M : B}\,(\multimap E)$$

Note that we used the notation $let\ x = N\ in\ M$ for $(\lambda x.M)N$ as introduced in Notation 3.4. □

We have thus recovered every rule of linear logic except for one, the $(R!)$ rule. And indeed we do not want it as part of our typing system. Note that using the constant *new* we can infer the valid judgement $\vdash new\, \mathfrak{o} : qbit$. Since the context is empty this would allow the application of the $(R!)$ rule resulting in a reusable qubit $\vdash new\ \mathfrak{o} : !qbit$ which we are trying to avoid at all costs. The rule however does hold in a special case as we will see at a later point.

3.1.4 Operational Semantics

With the language we have developed we can now write down quantum algorithms. The simplest of them is the *fair coin* defined as

$$\mathbf{coin} := \lambda \star .meas(H(new\, \mathfrak{o}))\,,$$

where H is the Hadamard gate. We can derive from our typing system the judgement $\vdash \mathbf{coin} : 1 \multimap !bit$. The fair coin outputs $\mathfrak{o}$ or $\mathfrak{1}$, each with a 50%

probability. Let's also define addition on bits.

$$\mathbf{plus} := \lambda xy.\, if\ x\ then\ (if\ y\ then\ \mathtt{0}\ else\ \mathtt{1})\ else\ (if\ y\ then\ \mathtt{1}\ else\ \mathtt{0})$$

Indeed we can derive $x : !bit, y : !bit \vdash \mathbf{plus} : !bit \otimes !bit \multimap !bit$. Now we can do addition with random numbers by executing the term

$$let\ x = \mathbf{coin} \star\ in\ \mathbf{plus}\, x\, x\,. \tag{3.4}$$

However we this leads to a problem. The result of this computation is ambiguous, it depends on the evaluation strategy. If we substitute $\mathbf{coin}\star$ directly into the addition we get $\mathbf{plus}(\mathbf{coin}\star)(\mathbf{coin}\star)$. This term can result in a $\mathtt{0}$ or a $\mathtt{1}$ with equal probability. This is the *call-by-name* evaluation strategy. If however we evaluate $\mathbf{coin}\star$ immediately and then enter the result into the addition, we obtain either $\mathbf{plus}\,\mathtt{0}\,\mathtt{0}$ or $\mathbf{plus}\,\mathtt{1}\,\mathtt{1}$ which both result in a $\mathtt{0}$. This is the *call-by-value* evaluation strategy. Clearly both strategies may yield different results. To remove this ambiguity we need to fix one of the methods, we chose the call-by-value strategie. Consequently we need to clarify which terms we are allowed to substitute and which need to be evaluated first.

Definition 3.17 (Values)**.** We define **values** to be the following terms

$$Value\ \ V, W\ \ ::=\ \ c\ \mid\ x\ \mid\ \lambda x.M\ \mid\ \langle V, W\rangle\ \mid\ \star\ \mid\ inj_l(V)\ \mid\ inj_r(V)$$

where M is an arbitrary term.

Convention 3.18. We only allow substitution by values. For a term M, a variable x and a value V we define $M[V/x]$ as in definition 3.3. We do however still allow the trivial substitution by terms in variables, i.e. $x[N/x]$ for arbitrary term N.

Definition 3.19 (Quantum Closure)**.** A **quantum closure**, or an **(operational) state** is a triple $[|Q\rangle, L, M]$ where

- $|Q\rangle$ is a quantum state of n qubits, i.e. a normalized vector in $(\mathbb{C}^2)^{\otimes n}$.
- L is a list of n distinct term variables $(x_1, .., x_n)$ with $x_i : qbit$, $i = 1, .., n$.
- M is a term with $FV(M) \subseteq \{x_1, .., x_n\}$.

We may also write $[Q, L, M]$ for $[|Q\rangle, L, M]$. Additionally we write $|L| = \{x_1, .., x_n\}$ for the (unordered) set of quantum variables, and $L(x_i) = i$ for the position of x_i in L. $[Q, L, V]$ is called a *value state* if V is a value. The notion of free variables can be extended to quantum closures.

$$FV([Q, L, M]) := FV(M) \backslash |L|.$$

Having a notion of free variables, α-equivalence extends to quantum closures. A quantum closure is evaluated via a series of reductions, the reduction rules are defined as follows.

Definition 3.20 (Reduction Rules)**.** We denote by $[Q, L, M] \to_p [Q', L', M']$ a single-step reduction that occurs with probability p. The reduction rules are

Table 3.3: Reduction rules for classical control of quantum closures. Here M, N are arbitrary terms and V, W are values.

$$
\begin{aligned}
[Q, L, \mathit{let}\ x = V\ \mathit{in}\ M] &\to_1 [Q, L, M[V/x]\ ] \\
[Q, L, \mathit{let}\ \langle x, y\rangle = \langle V, W\rangle\ \mathit{in}\ M] &\to_1 [Q, L, M[V/x, W/y]\ ] \\
[Q, L, \mathit{match}\ \mathit{inj}_l(V)\ \mathit{with}\ (x \mapsto M | y \mapsto N)] &\to_1 [Q, L, M[V/x]\ ] \\
[Q, L, \mathit{match}\ \mathit{inj}_l(V)\ \mathit{with}\ (x \mapsto M | y \mapsto N)] &\to_1 [Q, L, N[W/y]\ ]
\end{aligned}
$$

Table 3.4: Reduction rules for quantum data of quantum closures. Here U denotes a unitary gate of arity n. All closures are assumed to be in quantum state $|Q\rangle$. In the first rule the indices $j_i \in \{1, .., n\}$ are all distinct. We write $|Q^i_j\rangle$ for a superposition of states in which $x_i = |j\rangle$, j = 0,1, and α^i, β^i are such that $\alpha^i|Q^i_0\rangle + \beta^i|Q^i_1\rangle = |Q\rangle$.

$$
\begin{aligned}
[|Q\rangle, (x_1, .., x_n), U\langle x_{j_1}, .., x_{j_n}\rangle] &\to_1 [U|Q\rangle, (x_1, .., x_n), \langle x_{j_1}, .., x_{j_n}\rangle] \\
[\alpha^i|Q^i_0\rangle + \beta^i|Q^i_1\rangle, (x_1, .., x_n), \mathit{meas}\ x_i] &\to_{|\alpha^i|^2} [|Q^i_0\rangle, (x_1, .., x_n), \mathfrak{o}] \\
[\alpha^i|Q^i_0\rangle + \beta^i|Q^i_1\rangle, (x_1, .., x_n), \mathit{meas}\ x_i] &\to_{|\beta^i|^2} [|Q^i_0\rangle, (x_1, .., x_n), \mathfrak{1}] \\
[|Q\rangle, (x_1, .., x_n), \mathit{new}\ \mathfrak{o}] &\to_1 [|Q\rangle \otimes |0\rangle, (x_1, .., x_n, x_{n+1}), x_{n+1}] \\
[|Q\rangle, (x_1, .., x_n), \mathit{new}\ \mathfrak{1}] &\to_1 [|Q\rangle \otimes |1\rangle, (x_1, .., x_n, x_{n+1}), x_{n+1}]
\end{aligned}
$$

defined as shown in tables 3.3 and 3.4. Additionally we impose the congruence rules as given in table 3.5, and with it we completely fix the order of evaluation. We may write $\to$ for $\to_1$. To keep track of all the cases when evaluating the measurement we may also write

$$
[\ \alpha^i|Q^i_0\rangle + \beta^i|Q^i_1\rangle, (x_1, .., x_n), \mathit{meas}\ x_i\ ] \to \begin{cases} [\ |Q^i_0\rangle, (x_1, .., x_n), \mathfrak{o}\ ] \\ [\ |Q^i_1\rangle, (x_1, .., x_n), \mathfrak{1}\ ]\,. \end{cases}
$$

Example 3.21. We return to our two examples, the fair coin and adding bits (cmp. eq. (3.4)). Note that these terms do a priori not contain any quantum variables, so the according quantum closures have the form

$$
\begin{aligned}
&[\ |\rangle, (), \mathbf{coin}\star\ ] \\
&[\ |\rangle, (), \mathit{let}\ x = \mathbf{coin}\star\ \mathit{in}\ \mathbf{plus}\, x\, x\ ]\,.
\end{aligned}
$$

Table 3.5: Congruence rules for quantum closures. Here M, N are arbitrary terms while V is a value.

$$\frac{[Q,L,N] \to_p [Q',L',N']}{[Q,L,MN] \to_p [Q',L',MN']} \qquad \frac{[Q,L,M] \to_p [Q',L',M']}{[Q,L,MV] \to_p [Q',L',M'V]}$$

$$\frac{[Q,L,N] \to_p [Q',L',N']}{[Q,L,\langle M,N\rangle] \to_p [Q',L',\langle M,N'\rangle]} \qquad \frac{[Q,L,M] \to_p [Q',L',M']}{[Q,L,\langle M,V\rangle] \to_p [Q',L',\langle M',V\rangle]}$$

$$\frac{[Q,L,M] \to_p [Q',L',M']}{[Q,L,inj_l(M)] \to_p [Q',L',inj_l(M')]} \qquad \frac{[Q,L,M] \to_p [Q',L',M']}{[Q,L,inj_r(M)] \to_p [Q',L',inj_r(M')]}$$

$$\frac{[Q,L,M] \to_p [Q',L',M']}{[Q,L,match\ M\ with...] \to_p [Q',L',match\ M'\ with...]}$$

$$\frac{[Q,L,M] \to_p [Q',L',M']}{[Q,L,let\ \langle x,y\rangle = M\ in\ N] \to_p [Q',L',let\ \langle x,y\rangle = M'\ in\ N]}$$

The reduction works as follows. Here z denotes a fresh variable

$$\begin{aligned}
&[\ |\rangle, (), \mathbf{coin}\star\ ] \\
&= [\ |\rangle, (), (\lambda\star.meas(H(new\,\mathfrak{o})))\star\ ] \\
&= [\ |\rangle, (), let\ z = \star\ in\ meas(H(new\,\mathfrak{o}))\ ] \\
&\to [\ |\rangle, (), meas(H(new\,\mathfrak{o}))[\star/z]\ ] \\
&= [\ |\rangle, (), meas(H(new\,\mathfrak{o}))\ ] \\
&\to [\ |0\rangle, (q), meas(H\,q)\ ] \\
&\to [\ H|0\rangle, (q), meas\,q\ ] \\
&= [\ \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, (q), meas\,q\ ] \\
&\to \begin{cases} [\ |0\rangle, (q), \mathfrak{o}\ ] \\ [\ |1\rangle, (q), \mathfrak{1}\ ]\,. \end{cases}
\end{aligned}$$

Using this reduction we can now evaluate addition of a random bit with itself,

too.

$$
\begin{aligned}
& [\,|\rangle, (), let\ x = \mathbf{coin}\star\ in\ \mathbf{plus}\,x\,x\,] \\
= \ & [\,|\rangle, (), (\lambda x.(\mathbf{plus}\,x\,x))(\mathbf{coin}\star)\,] \\
\to \ & \begin{cases} [\,|0\rangle, (q), (\lambda x.(\mathbf{plus}\,x\,x))\,\mathtt{o}\,] \\ [\,|1\rangle, (q), (\lambda x.(\mathbf{plus}\,x\,x))\,\mathtt{1}\,] \end{cases} \\
= \ & \begin{cases} [\,|0\rangle, (q), let\ x = \mathtt{o}\ in\ \mathbf{plus}\,x\,x\,] \\ [\,|1\rangle, (q), let\ x = \mathtt{1}\ in\ \mathbf{plus}\,x\,x\,] \end{cases} \\
\to \ & \begin{cases} [\,|0\rangle, (q), \mathbf{plus}\,\mathtt{o}\,\mathtt{o}\,] \\ [\,|1\rangle, (q), \mathbf{plus}\,\mathtt{1}\,\mathtt{1}\,] \end{cases} \\
\to \ & \begin{cases} [\,|0\rangle, (q), let\ x = \mathtt{o}\ in\ (let\ y = \mathtt{o}\ in\ (if\ x\ then\ (if\ y\ then\ \mathtt{o}\ else\ \mathtt{1})\ else\ (if\ y\ then\ \mathtt{1}\ else\ \mathtt{o})))\,] \\ [\,|1\rangle, (q), let\ x = \mathtt{1}\ in\ (let\ y = \mathtt{1}\ in\ (if\ x\ then\ (if\ y\ then\ \mathtt{o}\ else\ \mathtt{1})\ else\ (if\ y\ then\ \mathtt{1}\ else\ \mathtt{o})))\,] \end{cases} \\
\to \ & \begin{cases} [\,|0\rangle, (q), let\ y = \mathtt{o}\ in\ (if\ \mathtt{o}\ then\ (if\ y\ then\ \mathtt{o}\ else\ \mathtt{1})\ else\ (if\ y\ then\ \mathtt{1}\ else\ \mathtt{o}))\,] \\ [\,|1\rangle, (q), let\ y = \mathtt{1}\ in\ (if\ \mathtt{1}\ then\ (if\ y\ then\ \mathtt{o}\ else\ \mathtt{1})\ else\ (if\ y\ then\ \mathtt{1}\ else\ \mathtt{o}))\,] \end{cases} \\
\to \ & \begin{cases} [\,|0\rangle, (q), if\ \mathtt{o}\ then\ (if\ \mathtt{o}\ then\ \mathtt{o}\ else\ \mathtt{1})\ else\ (if\ \mathtt{o}\ then\ \mathtt{1}\ else\ \mathtt{o})\,] \\ [\,|1\rangle, (q), if\ \mathtt{1}\ then\ (if\ \mathtt{1}\ then\ \mathtt{o}\ else\ \mathtt{1})\ else\ (if\ \mathtt{1}\ then\ \mathtt{1}\ else\ \mathtt{o})\,] \end{cases} \\
\to \ & \begin{cases} [\,|0\rangle, (q), if\ \mathtt{o}\ then\ \mathtt{1}\ else\ \mathtt{o}\,] \\ [\,|1\rangle, (q), if\ \mathtt{1}\ then\ \mathtt{o}\ else\ \mathtt{1}\,] \end{cases} \\
\to \ & \begin{cases} [\,|0\rangle, (q), \mathtt{o}\,] \\ [\,|1\rangle, (q), \mathtt{o}\,] \end{cases}
\end{aligned}
$$

As expected we obtain a zero no matter which state was measured.

More examples can be found in [22].

Definition 3.22 (Well Typedness, Program)**.** Let $[Q, (x_1, .., x_n), M]$ be a quantum closure. We call it **well-typed** of type A, if $x_1 : qbit, .., x_n : qbit \vdash M : A$ is a valid judgement. Such a quantum closure is also called a **program**.

Theorem 3.23 (Type safety)**.** *If a program $[Q, L, M]$ does through the reduction steps reach a state $[Q', L', M']$ which can not be further reduced, then $[Q', L', M']$ is a value state.*

Proof. This was proven in [25], section 7.2 there. □

3.1.5 Equational Logic

We have defined the terms of our language and explained what their supposed to mean, we've also introduced a typing system to ensure soundness of terms. However we have yet to make sure that our terms actually behave as expected. When we define e.g. a function $\lambda x.M$ and apply it to a value $(\lambda x.M)V$ that clearly is supposed to be equal to the substitution $M[V/x]$, since that is what we usually mean by function application. This was already hinted at when we introduced the notation $let\ x = V\ in\ M$, which reads as a substitution, for $(\lambda x.M)V$. We thus aim towards defining an equivalence, to ensure that the

terms behave in an appropriate manner. Since the equivalence may depend on the context, we define it on judgements rather then on terms. To be sure that we find all the equivalences, they have to be derived systematically. We obtain them by analyzing the interaction of introduction and corresponding elimination rules. Obviously applying an introduction rule followed by the according elimination rule should leave us with what we started with, this is known as *β-equivalence* or *β-reduction*. Dually in *η-equivalence* or *η-expansion* we impose that eliminating a term and then reintroducing it should leave the term unchanged. Depending on the connective *η-expansion* can also mean that introducing a second term and re-eliminating it should yield the term we started with. η-equivalence can also be considered a reduction depending on which way we read it. The β-reduction and η-expansion for type $\mathsf{1}$ are the following.

$$\dfrac{\dfrac{}{\vdash \star : \mathsf{1}}\,(\nabla^!) \qquad \Gamma \vdash M : A}{\Gamma \vdash \mathit{let}\ \star = \star\ \mathit{in}\ M : A}\,(1E) \qquad \longrightarrow^{\beta} \qquad \Gamma \vdash M : A \tag{3.5}$$

$$\Gamma \vdash M : \mathsf{1} \qquad \longrightarrow^{\eta} \qquad \dfrac{\Gamma \vdash M : \mathsf{1} \qquad \dfrac{}{\vdash \star : \mathsf{1}}\,(\nabla^!)}{\Gamma \vdash \mathit{let}\ \star = M\ \mathit{in}\ \star : \mathsf{1}}\,(1E) \tag{3.6}$$

For the $\oplus$-connective there are two β-reductions since there are two introduction rules

$$\dfrac{\dfrac{!\Delta, \Gamma \vdash V : A}{!\Delta, \Gamma \vdash \mathit{inj}_l(V) : A \oplus B}\,(\oplus I_1) \qquad !\Delta, \Theta, x : A \vdash M : C \qquad !\Delta, \Theta, y : B \vdash N : C}{!\Delta, \Gamma, \Theta \vdash \mathit{match}\ \mathit{inj}_l(V)\ \mathit{with}\ (x \mapsto M \mid y \mapsto N) : C}\,(\oplus E)$$

$$\longrightarrow^{\beta} \qquad !\Delta, \Gamma, \Theta \vdash M[V/x] : C \tag{3.7}$$

$$\dfrac{\dfrac{!\Delta, \Gamma \vdash V : A}{!\Delta, \Gamma \vdash \mathit{inj}_r(V) : A \oplus B}\,(\oplus I_2) \qquad !\Delta, \Theta, x : A \vdash M : C \qquad !\Delta, \Theta, y : B \vdash N : C}{!\Delta, \Gamma, \Theta \vdash \mathit{match}\ \mathit{inj}_r(V)\ \mathit{with}\ (x \mapsto M \mid y \mapsto N) : C}\,(\oplus E)$$

$$\longrightarrow^{\beta} \qquad \Gamma, \Theta \vdash N[V/y] : C \tag{3.8}$$

$$\Gamma \vdash P : A \oplus B \qquad \longrightarrow^{\eta}$$

$$\dfrac{\Gamma \vdash P : A \oplus B \qquad \dfrac{\dfrac{}{x : A \vdash x : A}}{x : A \vdash \mathit{inj}_l(x) : A \oplus B}\,(\oplus I_1) \qquad \dfrac{\dfrac{}{y : B \vdash y : B}}{y : B \vdash \mathit{inj}_r(y) : A \oplus B}\,(\oplus I_2)}{\Gamma \vdash \mathit{match}\ P\ \mathit{with}\ (x \mapsto \mathit{inj}_l(x) \mid y \mapsto \mathit{inj}_r(y)) : A \oplus B}\,(\oplus E) \tag{3.9}$$

For the implication we again have multiple reduction rules. One for the substitution by values, and one for the substitution $x[N/x]$ which we allow for arbitrary

terms.

$$\dfrac{\dfrac{!\Delta, \Gamma, x : A \vdash M : B}{!\Delta, \Gamma \vdash \lambda x.M : !^n(A \multimap B)}\,(\multimap I_{1/2}) \qquad !\Delta, \Theta \vdash V : A}{!\Delta, \Gamma, \Theta \vdash \ let\ x = V\ in\ M : B}\,(\multimap E)$$

$$\longrightarrow^{\beta} \qquad !\Delta, \Gamma, \Theta \vdash M[V/x] : B \tag{3.10}$$

$$\dfrac{\dfrac{\overline{x : A \vdash x : A}}{\vdash \lambda x.x : !^n(A \multimap A)}\,(\multimap I_{1/2}) \qquad \Gamma \vdash M : A}{\Gamma \vdash \ let\ x = M\ in\ x : A}\,(\multimap E) \quad \longrightarrow^{\beta} \quad \Gamma \vdash x[M/x] : A \tag{3.11}$$

$$\Gamma \vdash M : !^n(A \multimap B) \quad \longrightarrow^{\eta} \quad \dfrac{\dfrac{\Gamma \vdash M : !^n(A \multimap B) \qquad \overline{x : A \vdash x : A}}{\Gamma, x : A \vdash Mx : B}\,(\multimap E)}{\Gamma \vdash \lambda x.(Mx) : !^n(A \multimap B)}\,(\multimap I_{1/2}) \tag{3.12}$$

Since variables and constants do not have both introduction and elimination rules, the only remaining case is that of the $\otimes$-connective.

$$\dfrac{\dfrac{!\Delta, \Gamma_1 \vdash V_1 : A_1 \qquad !\Delta, \Gamma_2 \vdash V_2 : A_2}{!\Delta, \Gamma_1, \Gamma_2 \vdash \langle V_1, V_2 \rangle : A_1 \otimes A_2}\,(\otimes I) \qquad !\Delta, \Theta, x_1 : A_1, x_2 : A_2 \vdash M : A}{!\Delta, \Gamma_1\Gamma_2, \Theta \vdash \ let\ \langle x_1, x_2 \rangle = \langle V_1, V_2 \rangle\ in\ M : A}\,(\otimes E)$$

$$\longrightarrow^{\beta} \qquad !\Delta, \Gamma_1, \Gamma_2, \Theta \vdash M[V_1/x_1, V_2/x_2] : A \tag{3.13}$$

$$\Gamma \vdash M : A_1 \otimes A_2 \qquad \longrightarrow^{\eta}$$

$$\dfrac{\Gamma \vdash M : A_1 \otimes A_2 \qquad \dfrac{\overline{x_2 : A_2 \vdash x_2 : A_2} \qquad \overline{x_1 : A_1 \vdash x_1 : A_1}}{x_1 : A_1, x_2 : A_2 \vdash \langle x_1, x_2 \rangle : A_1 \otimes A_2}\,(\otimes I)}{\Gamma, \Theta \vdash \ let\ \langle x_1, x_2 \rangle = M\ in\ \langle x_1, x_2 \rangle : A_1 \otimes A_2}\,(\otimes E) \tag{3.14}$$

Based on these reduction rules we can now introduce an equivalence relation on judgements.

Definition 3.24 (Axiomatic equivalence)**.** The equivalence $\approx$ is defined as the smallest equivalence relation on typing judgements, which satisfies the axioms in tables 3.6 and 3.7. For a typing context Γ and $M, N : A$ we write $\Gamma \vdash M \approx N : A$ to write that $\Gamma \vdash M : A$ and $\Gamma \vdash N : A$ are equivalent. If context and type are non-specific or understood from circumstances we also use the simpler notation $M \approx N$. Additionally for a term $M[N]$ which contains subterm N we impose the congruence rule

$$\dfrac{N \approx N'}{M[N] \approx M[N']}$$

We take a look at rule ($\beta_{\multimap}$) from table 3.6 to make the notation clear. By

Table 3.6: Axiomatic equivalence: Rules from β-reduction and η-expansion. Here $\ominus$ denotes the symmetric difference of sets, i.e. $S \ominus S' := (S \cup S') \backslash (S \cap S')$.

$$
\begin{array}{ll}
(\beta_{\multimap}) & \mathit{let}\ x = V\ \mathit{in}\ M \approx M[V/x] \\
(\beta^2_{\multimap}) & \mathit{let}\ x = N\ \mathit{in}\ x \approx N \\
(\beta_{\otimes}) & \mathit{let}\ \langle x, y\rangle = \langle V, W\rangle\ \mathit{in}\ M \approx M[V/x, W/y] \\
(\beta_{\star}) & \mathit{let}\ \star = \star\ \mathit{in}\ M \approx M \\
(\eta_{\multimap}) & \lambda x.Vx \approx V \\
(\eta_{\otimes}) & \mathit{let}\ \langle x, y\rangle = N\ \mathit{in}\ \langle x, y\rangle \approx N \\
(\eta_{\star}) & \mathit{let}\ \star = N\ \mathit{in}\ \star \approx N \\
(\eta_{\oplus}) & \mathit{match}\ P\ \mathit{with}\ (x \mapsto \mathit{inj}_l(x) \mid y \mapsto \mathit{inj}_r(y)) \approx P
\end{array}
$$

$$
\begin{array}{ll}
(\beta^l_{\oplus}) & \Gamma, \Theta \vdash \mathit{match}\ \mathit{inj}_l(V)\ \mathit{with}\ (x \mapsto M \mid y \mapsto N) \approx M[V/x] : C \\
& \quad (\mathit{if}\ \Theta, y : !^n B \vdash N : C\ \mathit{is\ valid\ and} \\
& \quad (FV(N)\backslash\{y\}) \ominus (FV(M)\backslash\{x\})\ \mathit{contains\ reusable\ variables\ only})
\end{array}
$$

$$
\begin{array}{ll}
(\beta^r_{\oplus}) & \Gamma, \Theta \vdash \mathit{match}\ \mathit{inj}_r(V)\ \mathit{with}\ (x \mapsto M \mid y \mapsto N) \approx N[V/y] : C \\
& \quad (\mathit{if}\ \Theta, x : !^n A \vdash M : C\ \mathit{is\ valid\ and} \\
& \quad (FV(N)\backslash\{y\}) \ominus (FV(M)\backslash\{x\})\ \mathit{contains\ reusable\ variables\ only})
\end{array}
$$

Table 3.7: Axiomatic equivalence: Substitution rules. If in a term M the subterm $let\ x = P\ in\ N$ appears, we imply that x does not appear in the rest of the term M, e.g. in the case (let_1) if $\psi = \langle x, y\rangle$, then x, y do not appear in P. In the following φ and ψ may both stand for x, $\langle x, y\rangle$ and $\star$ (not respectively).

$$
\begin{aligned}
(let_{app}) &\quad let\ x = M\ in\ (let\ y = N\ in\ xy) \approx MN \\
(let_{\lambda}) &\quad let\ x = V\ in\ \lambda y.M \approx \lambda y.(let\ x = V\ in\ M) \\
(let_{\otimes}) &\quad let\ x = M\ in\ (let\ y = N\ in\ \langle x, y\rangle) \approx \langle M, N\rangle \\
(let_1) &\quad let\ \varphi = (let\ \psi = M\ in\ N)\ in\ P \approx let\ \psi = M\ in\ (let\ \varphi = N\ in\ P) \\
(let_2) &\quad let\ \varphi = V\ in\ (let\ \psi = W\ in\ M) \approx let\ \psi = W\ in\ (let\ \varphi = V\ in\ M)
\end{aligned}
$$

$$
\begin{aligned}
(let^0_{\oplus}) &\quad let\ x = P\ in\ (match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2) \approx \\
&\qquad\qquad match\ (let\ x = P\ in\ M)\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2)
\end{aligned}
$$

$$
\begin{aligned}
(let^1_{\oplus}) &\quad let\ x = P\ in\ (match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2) \approx \\
&\qquad\qquad match\ M\ with\ (y_1 \mapsto (let\ x = P\ in\ N_1) \mid y_2 \mapsto N_2)
\end{aligned}
$$

$$
\begin{aligned}
(let^2_{\oplus}) &\quad let\ x = P\ in\ (match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2) \approx \\
&\qquad\qquad match\ M\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto (let\ x = P\ in\ N_2))
\end{aligned}
$$

rule $(\beta_{\multimap})$ we mean to say that for arbitrary context Γ and type A we have

$$(\,\Gamma \vdash \mathit{let}\ x = V\ \mathit{in}\ M : A\,)\ \approx\ (\,\Gamma \vdash M[V/x] : A\,)\,.$$

Lemma 3.25. *Let φ be as in table 3.7, then the following equivalences are derivable*

$$(\alpha_{let}) \qquad \Gamma, y : A \vdash M : B\ \approx\ \Gamma, x : A \vdash \mathit{let}\ y = x\ \mathit{in}\ M : B$$
$$(let^2_{\otimes}) \qquad \langle \mathit{let}\ \varphi = M\ \mathit{in}\ N, V\rangle \approx \mathit{let}\ \varphi = M\ \mathit{in}\ \langle N, V\rangle\,.$$

Proof. The proof is done by case distinction [25]. □

Ultimately we want to consider typing judgements up to equivalence. However that makes the usage of values difficult, since for a value V we have for example

$$\mathit{let}\ x = V\ \mathit{in}\ x \approx V\,.$$

While V is a value, that is per definition not the case for $\mathit{let}\ x = V\ \mathit{in}\ W$. We thus introduce the notion of an *extended value*, which is closed under equivalence.

Definition 3.26 (Extended Values)**.** We define **extended values** to be the following terms

$$\begin{aligned} \mathit{ExtValue}\ \ E, F, F' \ ::=\ \ & V \mid \langle E, F\rangle \mid \mathit{inj}_l(E) \mid \mathit{inj}_r(E) \\ & \mid \mathit{let}\ x = E\ \mathit{in}\ F \mid \mathit{let}\ \star = E\ \mathit{in}\ F \\ & \mid \mathit{let}\ \langle x, y\rangle = E\ \mathit{in}\ F \\ & \mid \mathit{match}\ E\ \mathit{with}\ (x \mapsto F \mid y \mapsto F') \end{aligned}$$

where V is an arbitrary value.

Let's assume for now that substitution of a value into an extended value yields an extended value again. By inspection of the defining rules of the equivalence relation we can deduce, that each extended value is equivalent to other extended values only. We prove the assumption we made in lemma 3.35. The following lemma explains the relation between values and extended values.

Lemma 3.27. *Let E be an extended value and $\vdash E : A$ valid. Assume furthermore that no constants are of type $!^n(A \otimes B)$ or $!^n(A \oplus B)$. Then there is a value V, such that $\vdash E \approx V : A$.*

Proof. The proof is done by induction on the length of term E. The condition on constants is necessary to ensure that terms of type $!^n(B_1 \otimes B_2)$ and $!^n(B_1 \oplus B_2)$ have been introduced by rules $(\otimes I)$ and $(\oplus I)$ respectively. □

Note that if we were to need a constant of type $!^n(B_1 \otimes B_2)$, we could instead define two constants c_1, c_2 of types $!^n B_1$, $!^n B_2$ respectively and then consider $\langle c_1, c_2\rangle$ and if we were to need a constant of type $!^n(B_1 \oplus B_2)$, we could instead define a constant c of type $!^n B_1$ and then consider $\mathit{inj}_l(c)$.

For consistency reasons we want the axiomatic equivalence to preserve validity of judgements. Before we can show that, we have to prove some intermediate

lemmata. These lemmata analyze the form of judgements and what we can derive from them by other means then applying the typing rules. We have for example already mentioned, that the $(R!)$ rule from linear logic does not hold in general in our typing system, but that it does hold in a special case. We begin with the formulation and proof of that statement.

Lemma 3.28 ($(R!)$ rule for values)**.** *Let V be a value. If $!\Delta \vdash V : A$ is valid, so is $!\Delta \vdash V : !A$.*

Proof. The proof is done by structural induction on V. It is non-constructive. We begin with the base cases. Let in the following M be an arbitrary term.

- *Case $V \equiv x$* : The judgement $!\Delta \vdash x : A$ must have been introduced by (ax_1). There is thus a type B and a context Δ' such that $!\Delta = (!\Delta', x : !B)$ and $!B \leq: A$. By lemma 3.11 we thus also have $!B \leq: !A$ so that we can infer from rule (ax_1) that $\Delta', x : !B \vdash x : !A$ is valid.

- *Case $V \equiv c$* : The judgement $!\Delta \vdash c : A$ must have been introduced by (ax_2). We thus have that $!A_c \leq: A$. By lemma 3.11 also $!A_c \leq: !A$ so that we can infer from rule (ax_2) that $!\Delta \vdash c : !A$ is valid.

- *Case $V \equiv \star$* : It must be $A = !^n\mathbf{1}$, $n \in \mathbb{N}$, implying $!A = !^{n+1}\mathbf{1}$. From rule $(\nabla^!)$ it then follows that $!\Delta \vdash \star : !A$ is valid.

- *Case $V \equiv \lambda x.M$* : There must be types B, C such that $A = !^n(B \multimap C)$, $n \in \mathbb{N}$, and $!\Delta, x : B \vdash M : C$ is valid. Then $!A = !^{n+1}(B \multimap C)$. From rule $(\multimap I_2)$ it follows that $!\Delta \vdash \lambda x.M : !A$ is valid.

We continue with the induction steps. Assume that the lemma holds for values U, U_1, U_2.

- *Case $V \equiv inj_l(U)$* : There must be types B, C such that $A = !^n(B \oplus C)$, $n \in \mathbb{N}$, and $!\Delta \vdash U : !^n B$ is valid. By induction hypothesis $!\Delta \vdash U : !^{n+1} B$ is valid, too. Since $!A = !^{n+1}(B \oplus C)$ from rule $(\oplus I_1)$ the validity of $!\Delta \vdash inj_l(U) : !A$ follows.

- *Case $V \equiv inj_r(U)$* : Analogously.

- *Case $V \equiv \langle U_1, U_2 \rangle$* : There must be types A_1, A_2 such that $A = !^n(A_1 \otimes A_2)$, $n \in \mathbb{N}$, and valid judgements $!\Theta, !\Gamma_i \vdash U_i : !^n A_i$, $i = 1,2$, with $(!\Theta, !\Gamma_1, !\Gamma_2) = !\Delta$. By induction hypothesis the judgements $!\Theta, !\Gamma_i \vdash U_i : !^{n+1} A_i$ are valid, too. Since $!A = !^{n+1}(A_1 \otimes A_2)$ from rule $(\otimes I_1)$ the validity of $!\Delta \vdash \langle U_1, U_2 \rangle : !A$ follows.

That concludes the proof. □

Lemma 3.29. *Let $\Gamma \vdash M : A$ be valid, then $FV(M) \subseteq |\Gamma|$.*

Proof. This was proven in [25] (lemma 9.1.11 there). The proof works by structural induction over M, it is non-constructive. We will show the induction steps involving the coproduct, since these are not found in the source. So assume the lemma holds for terms P, N_1, N_2.

- *Case $M \equiv inj_l(P)$* : There must then be types A_1, A_2 and $n \in \mathbb{N}$ such that $A = !^n(A_1 \oplus A_2)$ and $\Gamma \vdash N : !^n A_1$ is valid. By induction hypothesis we then have $FV(N) \subseteq |\Gamma|$. The claim follows from $FV(M) = FV(N)$.

Case $M \equiv inj_r(P)$: Analogously.

Case $M \equiv$ *match* P *with* $(x_1 \mapsto N_1 \mid x_2 \mapsto N_2)$: The judgement can only have been introduced by rule $(\oplus E)$. There must thus be valid judgements $!\Delta, \Theta_1 \vdash P : !^n(B \oplus C)$, $!\Delta, \Theta_2, x_1 : !^n B \vdash N_1 : A$, $!\Delta, \Theta_2, x_2 : !^n C \vdash N_2 : A$ with $(!\Delta, \Theta_1, \Theta_2) = \Gamma$. By the induction hypothesis we may conclude that $FV(P) \subseteq |!\Delta, \Theta_1|$ and $FV(N_i) \subseteq |!\Delta, \Theta_2| \cup \{x_i\}$, $i = 1,2$. And since we have $FV(M) = FV(P) \cup (FV(N_1) \backslash \{x_1\}) \cup (FV(N_2) \backslash \{x_2\})$ it follows that $FV(M) \subseteq |!\Delta, \Theta_1| \cup |!\Delta, \Theta_2| = |\Gamma|$.

□

Lemma 3.30 (Generalized $(L!)$ rule)**.** *Let* $\Gamma, x : A \vdash M : B$ *be valid and* $A' \leq: A$*, then* $\Gamma, x : A' \vdash M : B$ *is valid, too.*

Proof. The proof is done by structural induction on M. It is non-constructive. We begin with the base cases.

Case $M \equiv y$: The judgement $\Gamma, x : A \vdash y : B$ must have been introduced by (ax_1). Thus we must have $y = x$, $\Gamma = !\Delta$ and $A \leq: B$. By transitivity the latter implies $A' \leq: B$ from which we can conclude by rule (ax_1) that $!\Delta, x : A' \vdash x : B$ is valid.

Case $M \equiv c$: The judgement must have been introduced by (ax_2). Thus there is context Δ and type A_1 such that $\Gamma = !\Delta$ and $A = !A_1$. By lemma 3.10 there is also A_1' such that $A' = !A_1'$. Then by rule (ax_2) $!\Delta, x : !A_1' \vdash c : B$ is valid.

Case $M \equiv \star$: The judgement must have been introduced by $(\nabla^!)$ and we thus have $B = !^n 1$, $\Gamma = !\Delta$ for some context Δ and $A = !A_1$ for some type A_1. By lemma 3.10 there then is a type A_1' such that $A' = !A_1'$. Now by $(\nabla^!)$ the validity of $!\Delta, x : !A_1' \vdash \star : !^n 1$ follows.

That concludes the base cases. For the induction steps assume that the lemma holds for terms N, P, N_1, N_2.

Case $M \equiv inj_l(N)$: The judgement must have been introduced by $(\oplus I_1)$ and there must thus be types C, D such that $B = !^n(C \oplus D)$ and $\Gamma, x : A \vdash N : !^n C$ is valid. By the induction hypothesis

Case $M \equiv inj_r(N)$: Analogously.

Case $M \equiv \lambda y.N$: The judgement must have been introduced by $(\multimap I_1)$ or $(\multimap I_2)$ and thus there is a type C such that $B = !^n(C \multimap D)$, $n \in \mathbb{N}$, and $\Gamma, x : A, y : C \vdash N : D$ is valid. By induction hypothesis $\Gamma, x : A', y : C \vdash N : D$ is then valid, too. If $n = 0$ we may simply apply $(\multimap I_1)$ and conclude that $\Gamma, x : A' \vdash \lambda y.N : B$ is valid. If $n > 0$ we have that $\Gamma = !\Delta$ for some context Δ and $A = !A_1$ for some type A_1. By lemma 3.10 there is A_1' such that $A' = !A_1'$. That allows us to apply $(\multimap I_2)$ and we can again conclude that $\Gamma, x : A' \vdash \lambda y.N : B$ is valid.

Case $M \equiv$ *match* P *with* $(y_1 \mapsto N_1 \mid y_2 \mapsto N_2)$: The judgement must have been introduced by $(\oplus E)$ and there must thus be valid judgements $!\Delta, \Theta_1 \vdash P : !^n(C \oplus D)$, $!\Delta, \Theta_1, y_1 : !^n C \vdash N_1 : B$ and $!\Delta, \Theta_1, y_2 : !^n D \vdash N_2 : B$

where $!\Delta, \Theta_1, \Theta_2$ are disjoint and $(!\Delta, \Theta_1, \Theta_2) = (\Gamma, x : A)$. We now have three cases: $x : A$ is in $!\Delta$, $x : A$ is in Θ_1 and $x : A$ is in Θ_2. Assume $x : A$ occurs in $!\Delta$. We then define $!\Delta'$ to be $!\Delta$ with $x : A$ replaced by $x : A'$, so that now $(!\Delta, \Theta_1, \Theta_2) = (\Gamma, x : A')$. By induction hypothesis the judgements $!\Delta', \Theta_1 \vdash P : !^n(C \oplus D)$, $!\Delta', \Theta_2, y_1 : !^n C \vdash N_1 : B$, $!\Delta', \Theta_2, y_2 : !^n D \vdash N_2 : B$ are valid and by $(\oplus E)$ so is $\Gamma, x : A' \vdash M : B$. If instead $x : A$ occurs in one of the Θ_i we proceed analogously.

The cases $M \equiv NP$, $M \equiv \langle N_1, N_2 \rangle$, $M \equiv$ *let* $\langle y_1, y_2 \rangle = N$ *in* P and $M \equiv$ *let* $\star = N$ *in* P are proven exactly like the last case. □

Lemma 3.31. *Let* $\Gamma \vdash V : !A$ *be valid with* V *a value. Then* $\Gamma = !\Delta$ *for some context* Δ*.*

Proof. This was proven in [25] (lemma 9.1.15 there). The proof is non-constructive. We show the induction steps in the cases of $V \equiv inj_l(W)$ and $V \equiv inj_r(W)$, since those can not be found in the source. So assume the lemma holds for value W.

Case $V \equiv inj_l(W)$: The judgement must have been introduced by rule $(\oplus I_1)$, thus there are types A_1, A_2 such that $!A = !(A_1 \oplus A_2)$ and $\Gamma \vdash W : !A_1$ is valid. By induction hypothesis we can conclude that $\Gamma = !\Delta$ for some context Δ.

Case $V \equiv inj_r(W)$: Analogously.

□

The restriction to values in above lemma is necessary. Indeed for a non-value term we can find a simple counterexample: obviously $y : A \multimap !B, x : A \vdash yx : !B$ is valid, but the context is not of the form $!\Delta$.

Lemma 3.32. *Let* $\Gamma, x : A \vdash M : B$ *be valid with* $x \notin FV(M)$*. The following are true.*

- $A = !A'$ *for some type* A'*;*
- $\Gamma \vdash M : B$ *is valid;*
- $\Gamma, x : A, y : !C \vdash M : B$ *is valid, for fresh* z*.*

Proof. This was proven in [25] (see lemma 9.1.19 there) by structural induction on M. We show here the induction steps which can not be found in the source. So let's assume the lemma holds true for terms P, N_1, N_2.

Case $M \equiv inj_l(P)$: The judgement must have been introduced by rule $(\oplus I)$. Thus there are types D_1, D_2 and $n \in \mathbb{N}$ such that $B = !^n(D_1 \oplus D_2)$ and $\Gamma, x : A \vdash P : !^n D_1$ is valid. Since $FV(inj_l(P)) = FV(P)$ we may conclude that $x \notin FV(P)$. By the induction hypothesis there is a type A' such that $A = !A'$. Also by induction hypothesis the judgements $\Gamma \vdash P : !^n D_1$ and $\Gamma, x : A, y : !C \vdash P : !^n D_1$ are valid. Then by rule $(\oplus I)$ so are $\Gamma \vdash inj_l(P) : !^n(D_1 \oplus D_2)$ and $\Gamma, x : A, y : !C \vdash inj_l(P) : !^n(D_1 \oplus D_2)$.

Case $M \equiv inj_r(P)$: Analogously.

Case $M \equiv$ *match* P *with* $(z_1 \mapsto N_1 \mid z_2 \mapsto N_2)$: The judgement must have been introduced by rule $(\oplus E)$. Thus there must be valid judgements $!\Delta, \Theta \vdash P : !^n(D_1 \oplus D_2)$ and $!\Delta, \Theta', z_i : !^n D_i \vdash N_i : B$, $i = 1{,}2$, with $(!\Delta, \Theta, \Theta') = (\Gamma, x : A)$. Since contexts are disjoint it is $z_1 \neq x \neq z_2$. From $FV(M) = FV(P) \cup (FV(N_1)\backslash\{z_1\}) \cup (FV(N_2)\backslash\{z_2\})$ it then follows that $x \notin FV(P)$ and $x \notin FV(N_i)$, $i = 1{,}2$. Thus no matter to which of $!\Delta, \Theta, \Theta'$ the typing $x : A$ belongs, we can apply the induction hypothesis to deduce, that $A = !A'$ for some A'. Now let $\Delta_1, \Theta_1, \Theta_1'$ be respectively Δ, Θ, Θ' with the possible occurence of $x : A$ removed, so that $(\Delta_1, \Theta_1, \Theta_1') = \Gamma$. By induction hypothesis $!\Delta_1, \Theta \vdash P : !^n(D_1 \oplus D_2)$ and $!\Delta, \Theta', z_i : !^n D_i \vdash N_i : B$, $i = 1{,}2$, are valid, and thus by rule $(\oplus E)$ so is $\Gamma \vdash M : B$. Also by induction hypothesis $!\Delta, \Theta, y : !C \vdash P : !^n(D_1 \oplus D_2)$ and $!\Delta, \Theta', y : !C, z_i : !^n D_i \vdash N_i : B$, $i = 1{,}2$, are valid and again by rule $(\oplus E)$ so is $\Gamma, x : A, y : !C \vdash M : B$.

The proof is non-constructive. □

Corollary 3.33 ((!w) for contexts)**.** *If* $\Gamma \vdash M : B$ *is valid, then so is* $!\Delta, \Gamma \vdash M : B$.

Proof. Repeated application of the previous lemma. □

The attentive reader will question why this result is necessary, since we have shown in the last chapter that rule $(\nabla^!)$ implies $(!w)$ (cmp. lemma 2.26). However if we translate that derivation into our typing system, we get the rule

$$\frac{\Gamma \vdash M : B}{\Gamma, x : !A \vdash N[M] : B}$$

with $N[M] \equiv$ *let* $w =$ (*let* $\langle y, z\rangle = \langle M, x\rangle$ *in* $\langle y, \star\rangle$) *in* (*let* $u = *$ *in* v)). Indeed using the equivalence relation we can see, that $\Gamma, x : !A \vdash N[M] \approx M : B$, however we have not shown yet, that the equivalence preserves validity, so it is not ready for employment. Since however we will need $(!w)$ for the proof that equivalence preserves validity, lemma 3.32 and corollary 3.33 are necessary. Indeed the system was set up for these results to hold already now. Note that the rules (ax_1), (ax_2) contain an additional context $!\Delta$, which ensure that $(!w)$ holds for the bases cases of the induction in the proof of lemma 3.32.

Lemma 3.34. *Let* $\Gamma \vdash M : A$ *be valid and* $A \leq: B$*, then* $\Gamma \vdash M : B$ *is valid, too.*

Proof. The proof is done by structural induction on the term M. It is non-constructive.

Case $M \equiv x$: The judgement must have been introduced by (ax_1), thus Γ must be of the form $(!\Delta, x : A')$ for some $A' \leq: A$. By transitivity of the subtyping relation we have $A' \leq: B$, too. From (ax_1) it then follows that $!\Delta, x : A' \vdash x : B$ is valid.

Case $M \equiv c$: The judgement must have been introduced by (ax_2), thus $\Gamma = !\Delta$ for some Δ and $!A_c \leq: A$. By transitivity we then have $!A_c \leq: B$. From (ax_2) it then follows that $!\Delta \vdash c : B$ is valid.

Case $M \equiv \star$: The judgement must have been introduced by $(\nabla^!)$, thus $\Gamma = !\Delta$ for some Δ and $A = !^n\mathbf{1}$ for some $n \in \mathbb{N}$. By definition of the subtyping relation we must then have $B = !^m\mathbf{1}$ for some $m \in \mathbb{N}$. From $(\nabla^!)$ it then follows that $!\Delta \vdash \star : B$ is valid.

That concludes the base cases, we continue with the induction steps.

Case $M \equiv inj_l(N)$: The judgement must have been introduced by $(\oplus I_1)$, thus $A = A_1 \oplus A_2$ for some types A_1, A_2 and $\Gamma \vdash N : A_1$ is valid. Since $A_1 \oplus A_2 \leq: B$ we can conclude from the definition of the subtyping relation, that $B = B_1 \oplus B_2$ for some types B_1, B_2 and $A_1 \leq: B_1$ as well as $A_2 \leq: B_2$. By the induction hypothesis from $\Gamma \vdash N : A_1$ being valid and $A_1 \leq: B_1$ follows that $\Gamma \vdash N : B_1$ is valid, too. From $(\oplus I_1)$ it then follows that $\Gamma \vdash inj_l(N) : B$ is valid.

Case $M \equiv inj_r(N)$: Analogous to the above case.

Case $M \equiv$ *match* P *with* $(x_1 \mapsto N_1 | x_2 \mapsto N_2)$: The judgement must have been introduced by $(\oplus E)$. Thus there must be valid judgements $!\Delta, \Theta_1 \vdash P : C \otimes D$, $!\Delta, \Theta_2, x : C \vdash N_1 : A$ and $!\Delta, \Theta_2, y : D \vdash N_2 : A$ such that $(!\Delta, \Theta_1, \Theta_2) = \Gamma$. By the induction hypothesis we my conclude that $!\Delta, \Theta_2, x : C \vdash N_1 : B$ and $!\Delta, \Theta_2, y : D \vdash N_2 : B$ are also valid. From $(\oplus E)$ it then follows that $\Gamma \vdash$ *match* P *with* $(x_1 \mapsto N_1 | x_2 \mapsto N_2)$ is valid.

Case $M \equiv \lambda x.N$: The judgement must have been introduced by $(\multimap I_1)$ or $(\multimap I_2)$. Thus there are $n \in \mathbb{N}$ and types C, D such that $x : C$, $N : D$ and $A = !^n(C \multimap D)$ and $\Gamma, x : C \vdash N : D$ is valid. By inspection of the subtyping rules there must then be types C', D' such that $B = C' \multimap D'$ and $C' \leq: C$, $D \leq D'$. By the induction hypothesis $\Gamma, x : C \vdash N : D'$ is valid, too. Applying the generalized $(L!)$ rule (lemma 3.30) $\Gamma, x : C' \vdash N : D'$ is also valid. From $(\multimap I_1)$ (if $n = 0$) or $(\multimap I_2)$ (if $n > 0$) it then follows that $\Gamma' \vdash \lambda x.N : B$ is valid.

Case $M \equiv NP$: The judgement must have been introduced by $(\multimap E)$. Thus there must be valid judgements $!\Delta, \Theta_1 \vdash N : C \multimap A$ and $!\Delta, \Theta_2 \vdash P : C$ with $(!\Delta, \Theta_1, \Theta_2) = \Gamma$. Since $C \multimap A \leq: C \multimap B$ we may deduce by the induction hypothesis that $!\Delta, \Theta_1 \vdash N : C \multimap B$ is valid, too. From $(\multimap E)$ it then follows that $\Gamma \vdash NP : B$ is valid.

Case $M \equiv \langle N_1, N_2 \rangle$: The judgement must have been introduced by $(\otimes I)$. Thus there must be valid judgements $!\Delta, \Theta_1 \vdash N_1 : !^n A_1$, $!\Delta, \Theta_2 \vdash N_2 : !^n A_2$ with $n \in \mathbb{N}$, $A = !^n(A_1 \otimes A_2)$ and $(!\Delta, \Theta_1, \Theta_2) = \Gamma$. By inspection of the subtyping rules there must thus be types B_1, B_2 and $m \in \mathbb{N}$ such that $A_i \leq B_i$, $i = 1,2$, and $B = !^m(B_1 \otimes B_2)$. From $(\otimes I)$ it then follows that $\Gamma \vdash \langle N_1, N_2 \rangle : B$ is valid.

Case $M \equiv$ *let* $\langle x_1, x_2 \rangle = N$ *in* P : The judgement must have been introduced by $(\otimes E)$. Thus there must be valid judgements $!\Delta, \Theta_1 \vdash N : !^n(A_1 \otimes A_2)$, $!\Delta, \Theta_2, x_1 : !^n A_1, x_2 : !^n A_2 \vdash N : A$ with $!\Delta, \Theta_1, \Theta_2) = \Gamma$. By induction hypothesis $!\Delta, \Theta_2, x_1 : !^n A_1, x_2 : !^n A_2 \vdash N : B$ is valid, too. From $(\otimes E)$ it then follows that $\Gamma \vdash$ *let* $\langle x_1, x_2 \rangle = N$ *in* $P : B$ is valid.

Case $M \equiv \ let \star = N \ in \ P$: The judgement must have been introduced by $(1E)$. Thus there must be valid judgements $!\Delta, \Theta_1 \vdash N : 1$ and $!\Delta, \Theta_2 \vdash P : A$ with $(!\Delta, \Theta_1, \Theta_2) = \Gamma$. By induction hypothesis $!\Delta, \Theta_2 \vdash P : B$ is valid, too. From $(1E)$ it then follows that $\Gamma \vdash let \star = N \ in \ P : B$ is valid.

By induction the claim follows. □

Lemma 3.35. *Let V be a value. If M is a value (resp. extended value), then so is $M[V/x]$.*

Proof. The proof is done by structural induction. The base cases are proven as follows:

$M \equiv c$: we have $M[V/x] = c[V/x] = c$ which is a value.

$M \equiv \star$: we have $M[V/x] = \star[V/x] = \star$ which is a value.

$M \equiv y$: if $x = y$ we have $M[V/x] = x[V/x] = V$, if instead $x \neq y$ we have $M[V/x] = y[V/x] = y$. Both are values.

$M \equiv \lambda y.N$: if $x = y$ we have $M[V/x] = (\lambda y.N)[V/x] = \lambda y.N$, if instead $x \neq y$ we have $M[V/x] = (\lambda y.N)[V/x] = \lambda y.(N[V/x])$. Both are values.

That proves the base case. The induction steps for values are the following.

$M \equiv \langle U, W \rangle$: We have $M[V/x] = \langle U, W \rangle[V/x] = \langle U[V/x], W[V/x] \rangle$. Hence if $U[V/x], W[V/x]$ are values, so is $M[V/x]$.

$M \equiv inj_l(U)$: It is $M[V/x] = inj_l(U)[V/x] = inj_l(U[V/x])$. Thus if $U[V/x]$ is a value, so is $M[V/x]$.

$M \equiv inj_r(U)$: Analogously.

By induction this finishes the proof for values. For extended values the base case is the case of M being a (non-extended) value, which we have just proven. We continue with the induction steps. Assume in the following that the lemma holds for extended values E, F, F_1, F_2.

$M \equiv \langle E, F \rangle$: We have $M[V/x] = \langle E, F \rangle[V/x] = \langle E[V/x], F[V/x] \rangle$. Hence if $E[V/x], F[V/x]$ are extended values, so is M.

$M = \ let \star = E \ in \ F$: We have $M[V/x] = (let \star = E \ in \ F)[V/x] = let \star = E[V/x] \ in \ F[V/x]$. Thus if $E[V/x], F[V/x]$ are extended values, so is $M[V/x]$.

$M \equiv \ let \ y = E \ in \ F$: by notational definition that means $M = (\lambda y.F)E$. If $x = y$ we have $M[V/x] = ((\lambda y.F)E)[V/x] = (\lambda y.F)(E[V/x])$. If instead $x \neq z$ we have $M[V/x] = ((\lambda y.F)E)[V/x] = (\lambda y.(F[V/x]))(E[V/x])$. Both are extended values if $F[V/x], E[V/x]$ are.

$M \equiv \ let \ \langle y, z \rangle = E \ in \ F$: If $x \in \{y, z\}$ we have $M[V/x] = \ let \ \langle y, z \rangle = E[V/x] \ in \ F$. If instead $x \notin \{y, z\}$ we have $M[V/x] = \ let \ \langle y, z \rangle = E[V/x] \ in \ F[V/x]$. Both are extended values if $F[V/x], E[V/x]$ are.

$\underline{M = \ match\ E\ with\ (y_1 \mapsto F_1 \mid y_2 \mapsto F_2)}$: If $y_1 \neq x \neq y_2$ we have

$$M[V/x] = match\ E[V/x]\ with\ (y_1 \mapsto F_1[V/x] \mid y_2 \mapsto F_2[V/x])\,,$$

so if $E[V/x], F_1[V/x], F_2[V/x]$ are extended values, so is $M[V/x]$. If instead $x = y_1$ it is

$$M[V/x] = match\ E[V/x]\ with\ (y_1 \mapsto F_1 \mid y_2 \mapsto F_2[V/x])\,,$$

so if $E[V/x], F_2[V/x]$ are extended values, so is $M[V/x]$. Analogously if $x = y_2$.

By induction that finishes the proof. □

Lemma 3.36 (Substitution Lemma)**.** *Assume that the two judgements*

$$!\Delta, \Gamma_1 \vdash V : A \qquad\qquad !\Delta, \Gamma_2, x : A \vdash M : B \tag{3.15}$$

are valid, with $|\Gamma_1| \cap |\Gamma_2| = \emptyset$ *and with* V *a value. Then the following is valid too*

$$!\Delta, \Gamma_1, \Gamma_2 \vdash M[V/x] : B\,. \tag{3.16}$$

Proof. This was proven in [25]. The proof is done by structural induction over M, it is non-constructive. We only show the induction steps involving the coproduct, since these can not be found in the literature. So assume the lemma holds for terms P, N, N_1, N_2.

$\underline{Case\ M \equiv inj_l(N)}$: Since $!\Delta, \Gamma_2, x : A \vdash inj_l(N) : B$ is valid, there must be a types B_1, B_2, such that $B = !^n(B_1 \oplus B_2)$, $n \in \mathbb{N}$, and such that $!\Delta, \Gamma_2, x : A \vdash N : !^n B_1$ is valid. By the induction hypothesis $!\Delta, \Gamma_1, \Gamma_2, x : A \vdash N[V/x] : !^n B_1$ is valid, too. Applying rule $(\oplus I_1)$ we get $!\Delta, \Gamma_1, \Gamma_2, x : A \vdash inj_l(N[V/x]) : B$. The statement now follows from $inj_l(N[V/x]) = inj_l(N)[V/x]$.

$\underline{Case\ M \equiv inj_l(N)}$: Analogously.

$\underline{Case\ M \equiv match\ P\ with\ (y_1 \mapsto N_1 \mid y_2 \mapsto N_2)}$: There must be valid judgements $!\Delta', \Theta \vdash P : !^n(B_1 \oplus B_2)$, $!\Delta', \Theta', y_1 : !^n B_1 \vdash N_1 : B$ and $!\Delta', \Theta', y_2 : !^n B_2 \vdash N_2 : B$ with $(!\Delta', \Theta, \Theta') = (!\Delta, \Gamma_2, x : A)$. Since we consider contexts to be disjoint we have in particular $y_1 \neq x \neq y_2$ and substitution thus works the following way

$$M[V/x] = match\ P[V/x]\ with\ (y_1 \mapsto N_1[V/x] \mid y_2 \mapsto N_2[V/x]).$$

We have three cases to consider.
If $x : A$ occurs in context $!\Delta'$ there is type A' such that $A = !A'$. By lemma 3.31 there is then Γ_1' such that $\Gamma_1 = !\Gamma_1'$. Let $!\tilde{\Delta}'$ be $!\Delta'$ without the typing of x. Now we have to do a little bit of context-yoga to avoid overlapping contexts in our judgements. So let $!\tilde{\Delta}_1', !\Theta_1, !\Theta_1'$ be the parts of $!\tilde{\Delta}', \Theta, \Theta'$ (respectively) which also occur in $!\Delta$. The other parts, which

then also occur in Γ_2, are denoted by $!\tilde{\Delta}'_2, \Theta_2, \Theta'_2$. Let furthermore $!\Lambda, !\Lambda'$ be such that $(!\tilde{\Delta}'_1, !\Theta_1, !\Lambda) = !\Delta$ and $(!\tilde{\Delta}'_1, !\Theta'_1, !\Lambda') = !\Delta$. We then have $|!\tilde{\Delta}'_2, \Theta_2| \cap |!\Lambda, !\Gamma'_1| = \emptyset$ and $|!\tilde{\Delta}'_2, \Theta'_2| \cap |!\Lambda', !\Gamma'_1| = \emptyset$ and we may thus in each case apply the induction hypothesis which yields the judgements

$$!\tilde{\Delta}'_1, !\Theta_1, !\tilde{\Delta}'_2, \Theta_2, !\Lambda, !\Gamma'_1 \vdash P[V/x] : !^n(B_1 \oplus B_2)\,,$$

$$!\tilde{\Delta}'_1, !\Theta'_1, !\tilde{\Delta}'_2, \Theta'_2, !\Lambda', !\Gamma'_1, y_1 : !^n B_1 \vdash N_1[V/x] : B\,,$$

$$!\tilde{\Delta}'_1, !\Theta'_1, !\tilde{\Delta}'_2, \Theta'_2, !\Lambda', !\Gamma'_1, y_2 : !^n B_2 \vdash N_2[V/x] : B\,.$$

Note that all contexts that the first judgement has in common with the other two are reusable, so that we may apply rule $(\oplus E)$ from which we get

$$!\tilde{\Delta}'_1, !\tilde{\Delta}'_2, !\Gamma'_1, !\Theta_1, \Theta_2, !\Lambda, !\Theta'_1, \Theta'_2, !\Lambda' \vdash M[V/x] : B\,.$$

The context is by construction equal to $(!\Delta, \Gamma_1, \Gamma_2)$.
Let's now consider the case where the typing $x : A$ appears in context Θ. By lemma 3.29 that implies that x is not free in N_1, N_2 and thus $N_i[V/x] = N_i$, $i = 1{,}2$. Let $\tilde{\Theta}$ be Θ with the typing of x removed. Let $!\Delta'_1, !\tilde{\Theta}_1$ be the parts of $!\Delta', \tilde{\Theta}$ (respectively) that also occur in $!\Delta$ and $!\Delta'_2, \tilde{\Theta}_2$ the other parts, which then also occur in Γ_2. Let Λ be such that $(!\Delta'_1, !\tilde{\Theta}_1) = !\Delta$. Then we may apply the induction hypothesis to the judgement involving P and obtain

$$!\Delta'_1, !\tilde{\Theta}_1, !\Delta'_2, \tilde{\Theta}_2, \Gamma_1 \vdash P[V/x] : !^n(B_1 \oplus B_2)\,.$$

We can now apply rule $(\oplus E)$ to this and $!\Delta', \Theta', y_1 : !^n B_1 \vdash N_1 : B$, $!\Delta', \Theta', y_2 : !^n B_2 \vdash N_2 : B$, using $(!\Delta'_1, !\Delta'_2) = !\Delta'$ and we obtain

$$!\Delta', !\tilde{\Theta}_1, \tilde{\Theta}_2, \Gamma_1, \Theta' \vdash M[V/x] : B\,.$$

The context is by construction equal to $(!\Delta, \Gamma_1, \Gamma_2)$.
The case were $x : A$ occurs in context Θ' works analogously.

By induction that finishes the proof. □

The substitution lemma is sufficient to show, that β- and η-reduction preserves validity of judgements. However since we defined an equivalence relation from these, we need to argue, that the according expansions preserve validity, too. Otherwise a valid judgement could be in the same equivalence class as an invalid expanded form. We thus show a reverse substitution lemma, too.

Lemma 3.37 (Reverse Substitution Lemma)**.** *Let M be a term, V a value and $x \in FV(M)$. Assume that $\Theta \vdash M[V/x]$ is valid and let A be the type of V in that judgement. Then the following are valid:*

$$!\Delta, \Gamma_1 \vdash V : A \qquad\qquad !\Delta, \Gamma_2, x : A \vdash M : B$$

with $(!\Delta, \Gamma_1, \Gamma_2) = \Theta$.

Proof. We prove this by structural induction over the term M. The proof is non-constructive. Let's begin with the base cases.

Case $M \equiv y$: Since $x \in FV(M)$ we have $y = x$, $A = B$ and $M[V/x] = V$. Thus $\Theta \vdash V : B$ is valid, and so is $x : B \vdash x : B$. We are done.

Case $M \equiv c$: We have $x \in FV(c) = \emptyset$. This case can thus never occur.

Case $M \equiv \star$: We have $x \in FV(\star) = \emptyset$. This case can thus never occur.

We continue with the induction steps. For that we assume, that the lemma holds for terms P, N, N_1, N_2.

Case $M \equiv inj_l(N)$: There are types B_1, B_2 and $n \in \mathbb{N}$ such that $B = !^n(B_1 \otimes B_2)$ and $\Theta \vdash N[V/x] : !^n B_1$ is valid. Since we have $x \in FV(M) = FV(N)$ we can apply the induction hypothesis. Thus there are valid $!\Delta, \Gamma_1 \vdash V : A$ and $!\Delta, \Gamma_2, x : A \vdash N : !^n B_1$ with $(!\Delta, \Gamma_1, \Gamma_2) = \Theta$. Applying $(\oplus I)$ to the latter judgement, we get that $!\Delta, \Gamma_2, x : A \vdash inj_l(N) : !^n(B_1 \otimes B_2)$.

Case $M \equiv NP$: We have $FV(M) = FV(N) \cup FV(P)$ and $M[V/x] = N[V/x]P[V/x]$. There must then be valid judgements $!\Delta, \Gamma \vdash N[V/x] : C \multimap B$ and $!\Delta, \Gamma' \vdash P[V/x] : C$ with $(!\Delta, \Gamma, \Gamma') = \Theta$.
If $x \in FV(N) \cap FV(P)$ then there is A' such that $A = !A'$. By applying the induction hypothesis to $!\Delta, \Gamma \vdash N[V/x] : C \multimap B$ we obtain valid judgements $!\Delta', !\Gamma_1 \vdash V : !A'$ and $!\Delta', \Gamma_2, x : !A' \vdash N : C \multimap B$, with $(!\Delta', !\Gamma_1, \Gamma_2) = (!\Delta, \Gamma)$, where we already applied the $(L!)$ rule (lemma 3.30) to the former judgement. By applying the induction hypothesis to $!\Delta, \Gamma' \vdash P[V/x] : C$ we obtain valid judgements $!\Delta'', !\Gamma_1' \vdash V : !A'$ and $!\Delta'', \Gamma_2', x : !A' \vdash P : C$, with $(!\Delta'', !\Gamma_1', \Gamma_2') = (!\Delta, \Gamma')$, where we already applied the $(L!)$ rule (lemma 3.30) to the former judgement. We can w.l.o.g. assume that $!\Delta' = !\Delta''$ and $(!\Delta'', !\Gamma_1') = (!\Delta', !\Gamma_1)$ since otherwise the contexts may be adjusted by lemma 3.32. Thus applying the $(\multimap E)$ rule, we obtain $!\Delta', \Gamma_2, \Gamma_2', x : !A' \vdash NP : B$. It is $(!\Delta', !\Gamma_1, \Gamma_2, \Gamma_2') = (!\Delta, \Gamma, \Gamma') = \Theta$.
If $x \in FV(N) \backslash FV(P)$ we have $P[V/x] = P$ and thus $!\Delta, \Gamma' \vdash P[V/x] : C$. By induction hypothesis $!\Delta', \Gamma_1 \vdash V : A$ and $!\Delta', \Gamma_2, x : A \vdash N : C \multimap P$ are valid, with $(!\Delta', \Gamma_1, \Gamma_2) = (!\Delta, \Gamma)$. Applying $(\multimap E)$ then yields $!\Delta', \Gamma_2, \Gamma', x : A \vdash NP : B$. We also have $(!\Delta', \Gamma_1, \Gamma 2, \Gamma') = \Theta$.
The case $x \in FV(N) \backslash FV(P)$ works analogously.

Case $M \equiv \langle N_1, N_2 \rangle$: Analogously to previous case, since the assumptions of rules $(\otimes I)$ and $(\multimap E)$ are almost identical.

Case $M \equiv \lambda y.N$: We have $B = !^n(C \multimap D)$ for some types C, D. Since $x \in FV(M) = FV(N) \backslash \{y\}$ it is $x \neq y$. We assume without loss of generality that $y \notin FV(V)$, since otherwise we can rename y yielding an α-equivalent term. It is $M[V/x] = \lambda y.(N[V/x])$ and thus $\Theta, y : C \vdash N[V/x]$ is valid. By induction hypothesis there are valid judgements $!\Delta, \Gamma_1 \vdash V : A$ and $!\Delta, \Gamma_2, x : A \vdash N : D$ with $(!\Delta, \Gamma_1, \Gamma_2) = (\Theta, y : C)$. Since $y \in FV(N)$ the typing $y : C$ must appear in either Γ_2 or $!\Delta$.
If the typing $y : C$ appears in the context Γ_2 we define Γ_2' to be Γ_2 with $y : C$ removed. Then by rule $(\multimap I_{1\ or\ 2})$ $!\Delta, \Gamma_2', x : A \vdash \lambda y.N : !^n(C \multimap D)$ with $(\Delta, \Gamma_2', \Gamma_1) = \Theta$.
If instead $y : C$ appears in $!\Delta$, then we have a type C' with $C = !C'$. Let $!\Delta'$ be $!\Delta$ with $y : C$ removed. Then we have $!\Delta', \Gamma_1, y : !C' \vdash V : A$

and $!\Delta', \Gamma_2, x : A, y :!C' \vdash N : D$. Since $y \notin FV(V)$ we may drop $y :!C'$ from the former judgement due to lemma 3.32. To the latter judgement we apply $(\multimap I_{1\ or\ 2})$ and obtain $!\Delta', \Gamma_2, x : A \vdash \lambda y.N : D$. It is $(!\Delta', \Gamma_1, \Gamma_2) = \Theta$.

Case $M \equiv$ *let* $\langle y_1, y_2\rangle = N$ *in* P : It is

$$FV(M) = FV(N) \cup FV(P)\backslash\{y_1, y_2\}.$$

If $x \in FV(N) \cap FV(P)\backslash\{y_1, y_2\}$, in particular $y_1 \neq x \neq y_2$, we have $M[V/x] =$ *let* $\langle y_1, y_2\rangle = N[V/x]$ *in* $P[V/x]$ and $A = !A'$. There must thus be valid judgements $!\Delta, \Gamma \vdash N[V/x] :!^n(B_1 \otimes B_2)$, $!\Delta, \Gamma', y_1 :!^n B_1, y_2 :!^n B_2 \vdash P[V/x] : B$ with $(!\Delta, \Gamma, \Gamma') = \Theta$. Since x appears twice there must be a type A' such that $A = !A'$. Applying the induction hypothesis to the former judgement yields $!\Delta', !\Gamma_1 \vdash V :!A'$ and $!\Delta', \Gamma_2, x :!A' \vdash N :!^n(B_1 \otimes B_2)$, with $(!\Delta', !\Gamma_1, \Gamma_2) = (!\Delta, \Gamma)$, where we already applied the $(L!)$ rule (lemma 3.30) to the judgement involving V. Applying the induction hypothesis to $!\Delta, \Gamma', y_1 :!^n B_1, y_2 :!^n B_2 \vdash P[V/x] : B$ we obtain $!\Delta'', !\Gamma_1' \vdash V :!A'$ and $!\Delta'', \Gamma_2', x :!A' \vdash P : B$, with $(!\Delta'', !\Gamma_1', \Gamma_2') = (!\Delta, \Gamma', y_1 :!^n B_1, y_2 :!^n B_2)$. Without loss of generality we assume that the typings $y_1 :!^n B_1, y_2 :!^n B_2$ appear in Γ_2' or $!\Delta''$ (otherwise they are reusable and we can add them by means of lemma 3.32). Let $!\tilde{\Delta}'', \tilde{\Gamma}_2'$ be $!\Delta'', \Gamma_2'$ respectively with the (possibly occuring) typings of y_1, y_2 removed. Furthermore w.l.o.g. we assume $!\Delta' = !\tilde{\Delta}''$ and $!\Gamma_1 = !\Gamma_1'$, otherwise the contexts can be adjusted by lemma 3.32. Then by rule $(\otimes E)$ we can infer $!\Delta', \Gamma_2, \tilde{\Gamma}_2', x :!A' \vdash$ *let* $\langle y_1, y_2\rangle = N$ *in* $P : B$. It is $(!\Delta', !\Gamma_1, \Gamma_2, \tilde{\Gamma}_2') = (!\Delta, \Gamma, \Gamma') = \Theta$.
If instead $x \in (FV(P)\backslash\{y_1, y_2\})\backslash FV(N)$ we have $y_1 \neq x \neq y_2$ and $N[V/x] = N$ and $M[V/x] =$ *let* $\langle y_1, y_2\rangle = N$ *in* $P[V/x]$. There must thus be valid judgements $!\Delta, \Gamma \vdash N :!^n(B_1 \otimes B_2)$, $!\Delta, \Gamma', y_1 :!^n B_1, y_2 :!^n B_2 \vdash P[V/x] : B$ with $(!\Delta, \Gamma, \Gamma') = \Theta$. By induction hypothesis we then also have $!\Delta', \Gamma_1' \vdash V :!A'$ and $!\Delta', \Gamma_2', x : A \vdash P : B$, with $(!\Delta', \Gamma_1', \Gamma_2') = (!\Delta, \Gamma', y_1 :!^n B_1, y_2 :!^n B_2)$. We can again w.l.o.g. assume, that the typings of y_1, y_2 appear in Γ_2' (otherwise they are reusable and the contexts can be added if necessary) and we define $\tilde{\Gamma}_2'$ to be Γ_2' with the typings of y_1, y_2 removed. Then by rule $(\otimes E)$ we can infer $!\Delta', \Gamma, \tilde{\Gamma}_2', x : A \vdash$ *let* $\langle y_1, y_2\rangle = N$ *in* $P : B$. It is $(!\Delta', \Gamma, \Gamma_1', \tilde{\Gamma}_2') = (!\Delta, \Gamma', \Gamma) = \Theta$.
If instead $x \in FV(N)\backslash(FV(P)\backslash\{y_1, y_2\})$ then

$$M[V/x] = \textit{let } \langle y_1, y_2\rangle = N[V/x] \textit{ in } P.$$

There must thus be valid judgements $!\Delta, \Gamma \vdash N[V/x] :!^n(B_1 \otimes B_2)$, $!\Delta, \Gamma', y_1 :!^n B_1, y_2 :!^n B_2 \vdash P : B$ with $(!\Delta, \Gamma, \Gamma') = \Theta$. By the condition of disjoint contexts we must also have $x \notin \{y_1, y_2\}$ and thus $x \notin FV(P)$. We can simply apply the induction hypothesis and then rule $(\otimes E)$ and we are done.

Case $M \equiv$ *let* $\star = N$ *in* P : Like the previous case, but simpler.

Case $M \equiv$ *match* P *with* $(y_1 \mapsto N_1 \mid y_2 \mapsto N_2)$: This is again analogous to the case were $M \equiv$ *let* $\langle y_1, y_2\rangle = N$ *in* P, since the $(\oplus E)$ and $(\otimes E)$ rules have similar assumptions.

□

Theorem 3.38. *The axiomatic equivalence preserves validity of judgements.*

Proof. To prove that the equivalence preserves validity, we show that each of its defining rules does. The proof is non-constructive. We start with the rules obtained from β- and η-equivalence (table 3.6). From those we first consider the ones that do not involve any substitution.

Case $(\beta_\star)$: We need to show that $\Gamma \vdash \ let \star = \star \ in \ M : A$ is valid if and only if $\Gamma \vdash M : A$ is. For that we look at the β-reduction from which we infered this demand, namely

$$\dfrac{\dfrac{}{\vdash \star : 1}\,(\nabla^!) \qquad \Gamma \vdash M : A}{\Gamma \vdash \ let \star = \star \ in \ M : A}\,(1E) \qquad \longrightarrow^{\beta} \qquad \Gamma \vdash M : A\,.$$

Clearly if $\Gamma \vdash M : A$ is valid, so is $\Gamma \vdash \ let \star = \star \ in \ M : A$ by the derivation tree on the left of the reduction. Now assume that $\Gamma \vdash \ let \star = \star \ in \ M$ is valid. The last rule applied in its derivation can only have been $(1E)$ and thus there are contexts Δ, Θ with $(!\Delta, \Theta) = \Gamma$, such that $!\Delta \vdash \star : 1$ and $\Theta \vdash M : A$ are valid. Since $\Theta \vdash M : A$ is valid and $(!\Delta, \Theta) = \Gamma$ we deduce by $(!w)$ (corollary 3.33) that $\Gamma \vdash M : A$ is valid, too.

Case $(\eta_\star)$: Analogously.

Case $(\eta_\oplus)$: We need to show that $\Gamma \vdash P : A \oplus B$ is valid if and only if $\Gamma \vdash match \ P \ with \ (x \mapsto inj_l(x) \mid y \mapsto inj_r(y)) : A \oplus B$ is. Again if $\Gamma \vdash P : A \oplus B$ is valid, then by the derivation tree of the η-expansion (3.9) so is the other judgement. Lets assume now that $\Gamma \vdash match \ P \ with \ (x \mapsto inj_l(x) \mid y \mapsto inj_r(y)) : A \oplus B$ is valid. Then $(\oplus E)$ must have been applied, thus there are valid judgements $\Theta \vdash P : A \oplus B$, $\Theta', x : A \vdash inj_l(x) : A \oplus B$ and $\Theta', y : A \vdash inj_l(y) : A \oplus B$ with $(\Theta, \Theta') = \Gamma$. The judgement $\Theta', x : A \vdash inj_l(x) : A \oplus B$ can only have been derived from judgement $\Theta', x : A \vdash x : A$, which in turn must have been introduced by rule (ax_1). That implies that Θ' is actually of the form $!\Theta'$. Since then $(\Theta, !\Theta') = \Gamma$ and $\Theta \vdash P : A \oplus B$ is valid, it follows from $(!w)$ (corollary 3.33), that $\Gamma \vdash P : A \oplus B$ is valid, too.

The cases for $(\eta_{\multimap})$ and $(\eta_\otimes)$ can be handled analogously by tracing back the derivation tree of the expansion. So let's now look at the reductions that involve simple substitution.

Case $(\beta_{\multimap})$: We need to show that $\Gamma \vdash \ let \ x = V \ in \ M : B$ is valid if and only if $\Gamma \vdash M[V/x] : B$ is.
Assume the former judgement is valid. Then it must have been derived from judgements $!\Delta, \Gamma_1, x : A \vdash M : B$ and $!\Delta, \Gamma_2 \vdash V : A$ with $(!\Delta, \Gamma_1, \Gamma_2) = \Gamma$. By the substitution lemma 3.36 we can deduce that $\Gamma \vdash \ let \ x = V \ in \ M : B$ is valid.
Assume now, that $\Gamma \vdash \ let \ x = V \ in \ M : B$ is valid. Then by the reverse substitution lemma 3.37 we again obtain valid judgements $!\Delta, \Gamma_1, x : A \vdash M : B$ and $!\Delta, \Gamma_2 \vdash V : A$ with $(!\Delta, \Gamma_1, \Gamma_2) = \Gamma$. By the derivation tree in (3.10) we can then deduce that $\Gamma \vdash \ let \ x = V \ in \ M : B$ is valid.

Case $(\beta^2_{\multimap})$: We need to show that $\Gamma \vdash \ let\ x = M\ in\ x : A$ is valid if and only if $\Gamma \vdash \ x[M/x]$ is. Since $x[M/x] = M$ we basically have no substitution and this case can thus be handled like the cases $(\eta_\otimes)$ and $(\beta_\star)$ above.

Now we consider the reductions, that involve a more complicated substitution.

Case $(\beta^l_\oplus)$: We need to show that under the assumption $!\Delta, \Theta, y : !^n B \vdash N : C$ the judgement $\Gamma, \Theta \vdash M[V/x] : C$ is valid if and only if $\Gamma, \Theta \vdash match\ inj_l(V)\ with\ (x \mapsto M \mid y \mapsto N) : C$ is, where V is a value. Assume that the latter judgement is valid. It must have been introduced by rule $(\oplus E)$, so there are valid judgements $!\Delta, \Lambda \vdash inj_l(V) : !^n(A \oplus B)$, $!\Delta, \Lambda', x : !^n A \vdash M : C$ and $!\Delta, \Lambda', y : !^n B \vdash N : C$ with $(!\Delta, \Lambda, \Lambda') = (\Gamma, \Theta)$. In particular also $|\Lambda| \cap |\Lambda'| = \emptyset$. The judgement $!\Delta, \Lambda \vdash inj_l(V) : !^n(A \oplus B)$ must have been derived from assumption $!\Delta, \Lambda \vdash V : !^n A$. Now the substitution lemma 3.36 applies, and we may deduce that $\Gamma, \Theta \vdash M[V/x] : C$ is valid.
For the converse implication we assume that $\Gamma, \Theta \vdash M[V/x] : C$ is valid. Then by the reverse substituion lemma 3.37 we then have valid judgements $!\Delta, \Gamma_1 \vdash V : A$ and $!\Delta, \Gamma_2, x : A \vdash M : C$ with $(!\Delta, \Gamma_1, \Gamma_2) = (\Gamma, \Theta)$. Let Θ' be such that $(!\Delta, \Theta') = \Theta$. We then need to argue that we can demand $\Gamma_2 = \Theta'$. Excepting x, y, all non-reusable free variables that M, N have in common are in both Γ_2, Θ' and the ones they don't have in common must be reusable (this comes from the condition on free variables we have in the definition of $\beta^l_\oplus$). So we can by lemma 3.32 adjust Γ_2, Θ' to be equal. Then by the derivation tree of the $\beta_\oplus$ reduction (cmp. (3.7)) we have validity of $!\Delta, \Gamma_1, \Gamma_2 \vdash match\ inj_l(V)\ with\ (x \mapsto M \mid y \mapsto N) : C$ and $(!\Delta, \Gamma_1, \Gamma_2) = (\Gamma, \Theta)$.

Case $(\beta^r_\oplus)$: Analogously.

Case $(\beta_\otimes)$: We need to show that $\Gamma \vdash \ let\ \langle x_1, x_2\rangle = \langle V_1, V_2\rangle\ in\ M : B$ is valid if and only if $\Gamma \vdash M[V_1/x_1, V_2/x_2] : B$.
Assume the former judgement is valid. The there must be valid judgements $!\Delta, \Gamma_1 \vdash V_1 : A_1$, $!\Delta, \Gamma_2 \vdash V_2 : A_2$ and $!\Delta, \Theta, x_1 : A_1, x_2 : A_2 \vdash M : B$ with $(!\Delta, \Gamma_1, \Gamma_2, \Theta) = \Gamma$. Applying the substitution lemma 3.36 to the first and the third judgement we obtain $!\Delta, \Gamma_1, \Theta, x_2 : A_2 \vdash M[V_1/x_1] : B$. Applying the lemma again we obtain $!\Delta, \Gamma_1, \Gamma_2, \Theta \vdash M[V_1/x_1, V_2/x_2] : B$. Now let's assume that $\Gamma \vdash M[V_1/x_1, V_2/x_2] : B$ is valid. By the reverse substitution lemma 3.37 we obtain judgements $!\Delta, \Gamma_1, x_2 : A_2 \vdash M[V_1/x_1] : B$, $!\Delta, \Gamma_2 \vdash V_2 : A_2$ with $(!\Delta, \Gamma_1, \Gamma_2) = \Gamma$. Reverse substituting again we obtain $!\Delta', \Gamma_{12}, x_1 : A_1, x_2 : A_2 \vdash M : B$ and $!\Delta', \Gamma_{11} \vdash V_1 : A_1$ with $(!\Delta', \Gamma_{11}, \Gamma_{12}) = (!\Delta, \Gamma_1)$. By the derivation tree of (3.13) we can derive $\Gamma \vdash \ let\ \langle x_1, x_2\rangle = \langle V_1, V_2\rangle\ in\ M : B$.

That concludes the cases of β- and η-equivalence. The rules encompassing substitution (table 3.7) are all proven using the same simple argument. We show the proof for (let_{app}) . We need to show that $\Gamma \vdash MN : B$ is valid if and only if $\Gamma \vdash \ let\ x = M\ in\ (let\ y = N\ in\ xy) : B$ is. We show the derivation tree for the latter judgement. Remember that $let\ z = P_1\ in\ P_2$ is notation for

$(\lambda z.P_2)P_1$ and thus $let\ x = M\ in\ (let\ y = N\ in\ xy)$ means $(\lambda x.((\lambda y.(xy))N))M$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{!\Delta, x{:}A \multimap B \vdash x{:}A \multimap B} \quad \dfrac{}{!\Delta, y{:}A \vdash y{:}A}}{!\Delta, x : A \multimap B, y : A \vdash xy : B}}{!\Delta, x{:}A \multimap B \vdash \lambda y.(xy) : A \multimap B} \qquad !\Delta, \Gamma_2 \vdash N{:}A}{!\Delta, \Gamma_2, x : A \multimap B \vdash (\lambda y.(xy))N : B}}{!\Delta, \Gamma_2 \vdash \lambda x.((\lambda y.(xy))N) : (A \multimap B) \multimap B} \qquad !\Delta, \Gamma_1 \vdash M{:}A \multimap B}{!\Delta, \Gamma_1, \Gamma_2 \vdash (\lambda x.((\lambda y.(xy))N))M : B}$$

Note that there can be no fundamentally different derivation and that the two assumptions we made are sufficient to derive $!\Delta, \Gamma_1, \Gamma_2 \vdash MN : B$. Thus the two judgements can only be valid simultaneously, since any assumptions that derive the one, can also derive the other. All the other rules in table 3.7 involve two judgements for which the same property holds. Namely each assumption that proves the one, can also prove the other.

Lastly we should note, that if $\Gamma \vdash N : A$ and $\Gamma \vdash N' : A$ are only valid simultaneously, then in all occasions the same typing rules can be applied to the two judgements. Thus the congruence rule for our equivalence relation does preserve the property of simultaneous equivalence. That finishes the proof. □

With this result at hand, we can now safely consider judgements only up to axiomatic equivalence. That is crucial for the semantics which we consider now.

3.2 Categorical Semantics

In this section we analyze the categorical semantics of the *core fragment* of our type system.

3.2.1 Categories of Types

Definition 3.39 (Core Fragment)**.** The **core fragment** is:

$$\begin{aligned}
Type\ \ A, B &::= \alpha \mid\ !A \mid A \multimap B \mid 1 \mid A \otimes B \\
Value\ \ V, W &::= c \mid x \mid \lambda x.M \mid \langle V, W\rangle \mid \star \\
ExtValue\ \ E, F &::= V \mid \langle E, F\rangle \mid let\ x = E\ in\ F \mid let\ \langle x, y\rangle = E\ in\ F \\
Term\ \ M, N &::= V \mid MN \mid \langle M, N\rangle \mid let\ \langle x, y\rangle = M\ in\ N
\end{aligned}$$

where α and c are arbitrary but fixed sets of ground types and ground operations respectively.

Definition 3.40 (Category of Computations)**.**
The category of computations **Comp** is defined as follows:

- objects are types as in definition 3.39
- arrows $A \longrightarrow B$ are (equivalence classes of) valid typing judgements $x : A \vdash M : B$

- composition of $x : A \vdash M : B$ with $y : B \vdash N : C$ is $x : A \vdash let\ y = M\ in\ N : C$
- the identity id_A is $x : A \vdash x : A$.

For composition and identity note that these are valid judgements indeed. The identity being valid follows from rule (i), and validity of the composed arrow follows from the cut-rule. Composition is associative by equivalence rule (let_1). The identity actually being an identity is ensured by the derived rule (α_{let}) and by $(\beta^2_{-\circ})$.

Definition 3.41 (Category of Values)**.**
The category of values **Val** is defined such that:
objects are types as in definition 3.39
arrows $A \longrightarrow B$ are (valid) typing judgements $x : A \vdash E : B$ where E is an extended value
composition and identity are defined as in **Comp**.

Since extended values are closed under the axiomatic equivalence this is well defined.

Definition 3.42 (Category of classical Values)**.**
The category of classical values **cVal** is defined such that:
objects are types as defined in 3.39
arrows $A \longrightarrow B$ are (valid) typing judgements $x : !A \vdash E : !B$ where E is an extended value
composition and identity are defined as in **Comp**.

3.2.2 Product Structure

Lemma 3.43. $(\mathbf{Val}, \otimes)$ *is symmetric monoidal.*

Proof. The first steps of this proof will be carried out in a very explicit and rigorous manner, to give the reader an understanding of the workings, which then allows us to overgo some details afterwards.
Step 1: $\otimes$ is a functor $\otimes : \mathbf{Val} \times \mathbf{Val} \longrightarrow \mathbf{Val}$.

We already understand how $\otimes$ is acting on the objects. So now let $f : A \longrightarrow A'$, $g : B \longrightarrow B'$ be two arrows in **Val**, say $f = \ x : A \vdash V : A'$ and $g = \ y : B \vdash W : B'$. Then we define

$$f \otimes g := \ z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in\ \langle V, W\rangle : A' \otimes B'$$

which yields an arrow $f \otimes g : A \otimes B \longrightarrow A' \otimes B'$. Note that this corresponds to the $(\otimes)$-rule from symmetric monoidal theories (cmp. definition 2.21), the validity of the judgement $f \otimes g$ can thus be shown analogously as there (see theorem 2.25). By definition we have

$$\begin{aligned} id_A \otimes id_A &= z : A \otimes A \vdash \ let\ \langle x, x\rangle = z\ in\ \langle x, x\rangle : A \otimes A \\ &\approx z : A \otimes A \vdash z : A \otimes A \\ &= id_{A\otimes A}\,. \end{aligned}$$

For f, g as above and $f' : A' \longrightarrow A''$, $g' : B' \longrightarrow B''$, say $f' = x' : A' \vdash V'' : A''$ and $g' = y' : A' \vdash W'' : B''$ (with $x' \notin FV(W'')$, $y' \notin FV(V'')$), we have that

$$\begin{aligned}(f' \circ f) \otimes (g' \circ g) = &(x : A \vdash \ let\ x' = V\ in\ V'' : A'') \otimes \\ &(y : B \vdash \ let\ y' = W\ in\ W'' : B'') \\ = &z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\ &\langle let\ x' = V\ in\ V'',\ let\ y' = W\ in\ W''\rangle : A'' \otimes B'' \\ \overset{\beta_{\multimap}}{\approx} &z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\ &\langle V''[V/x'], W''[W/y']\rangle : A'' \otimes B'' ,\end{aligned}$$

while

$$\begin{aligned}(f' \otimes g') \circ (f \otimes g) = &(z' : A' \otimes B' \vdash \ let\ \langle x', y'\rangle = z'\ in\ \langle V'', W''\rangle : A'' \otimes B'') \circ \\ &(z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in\ \langle V, W\rangle : A' \otimes B') \\ = &z : A \otimes B \vdash \ let\ z' = (let\ \langle x, y\rangle = z\ in\ \langle V, W\rangle)\ in \\ &(let\ \langle x', y'\rangle = z'\ in\ \langle V'', W''\rangle) : A'' \otimes B'' \\ \overset{let_1}{\approx} &z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in\ (let\ z' = \langle V, W\rangle\ in \\ &(let\ \langle x', y'\rangle = z'\ in\ \langle V'', W''\rangle)) : A'' \otimes B'' \\ \overset{\beta_{\multimap}}{\approx} &z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\ &(let\ \langle x', y'\rangle = \langle V, W\rangle\ in\ \langle V'', W''\rangle) : A'' \otimes B'' \\ \overset{\beta_{\otimes}}{\approx} &z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\ &\langle V'', W''\rangle[V/x', W/y'] : A'' \otimes B'' \\ = &z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\ &\langle V''[V/x'], W''[W/y']\rangle : A'' \otimes B''\end{aligned}$$

where in the last step we used, that x' does not occur in W'' and y' not in V''. Thus $(f' \circ f) \otimes (g' \circ g) = (f' \otimes g') \circ (f \otimes g)$ and we my conclude, that $\otimes$ is indeed a functor.

Step 2: $(\mathbf{Val}, \otimes)$ is monoidal
To increase the readability of terms, we will from now on often denote a variable's type in a superscripted index. We need to identify a unit object I, an associator a and the left and right unitors l, r. We make the following choices:

$$I := 1 ,$$

$$\begin{aligned}a_{A,B,C} := \ &t : (A \otimes B) \otimes C \vdash \ let\ \langle u^{A\otimes B}, z^C\rangle = t\ in \\ &let\ \langle x^A, y^B\rangle = u^{A\otimes B}\ in\ \langle x^A, \langle y^B, z^C\rangle\rangle : A \otimes (B \otimes C) ,\end{aligned}$$

$$l_A := \; y : \mathbf{1} \otimes A \vdash \; let\ \langle t^{\mathbf{1}}, x^A \rangle = y\ in\ (let\ \star = t^{\mathbf{1}}\ in\ x^A) : A\,,$$
$$r_A := \; y : A \otimes \mathbf{1} \vdash \; let\ \langle x^A, t^{\mathbf{1}} \rangle = y\ in\ (let\ \star = t^{\mathbf{1}}\ in\ x^A) : A\,.$$

Of course it needs to be checked wether these judgements are actually valid in our typing system. This can be done by mimicking the derivations from the previous chapter. To derive for example the judgement l_A we carry out the derivation of rule (l) (see theorem 2.25)

$$\dfrac{\dfrac{}{y : \mathbf{1} \otimes A \vdash y : \mathbf{1} \otimes A}\,(i) \qquad \dfrac{\dfrac{}{t : \mathbf{1} \vdash t : \mathbf{1}}\,(\mathrm{i}) \qquad \dfrac{}{x : A \vdash x : A}\,(\mathrm{i})}{t : \mathbf{1}, y : Y \vdash \; let\ \star = t^{\mathbf{1}}\ in\ x^A : A}\,(\mathbf{1}E)}{y : \mathbf{1} \otimes A \vdash \; let\ \langle t^{\mathbf{1}}, x^A \rangle = y\ in\ (let\ \star = t^{\mathbf{1}}\ in\ x^A) : A}\,(\otimes E)$$

where we immediately transformed the derivation according to lemma 2.23. The inverses of above arrows are given by

$$l_A^{-1} := \; x : A \vdash \langle \star, x \rangle : \mathbf{1} \otimes A\,,$$
$$r_A^{-1} := \; x : A \vdash \langle x, \star \rangle : A \otimes \mathbf{1}\,.$$

$$a^{-1}_{A,B,C} := \; t : A \otimes (B \otimes C) \vdash \; let\ \langle x^A, u^{B \otimes C} \rangle = t\ in$$
$$let\ \langle y^B, z^C \rangle = u^{B \otimes C}\ in\ \langle \langle x^A, y^B \rangle, z^C \rangle : (A \otimes B) \otimes C\,.$$

We need to show they actually are inverses, indeed

$$\begin{aligned}
l_A \circ l_A^{-1} &= (y : \mathbf{1} \otimes A \vdash \; let\ \langle \star, x^A \rangle = y\ in\ x^A : A) \circ (z : A \vdash \langle \star, z \rangle : \mathbf{1} \otimes A) \\
&= x : A \vdash \; let\ y^{\mathbf{1} \otimes A} = \langle \star, x \rangle\ in \\
&\qquad\qquad let\ \langle t^{\mathbf{1}}, z^A \rangle = y\ in\ (let\ \star = t^{\mathbf{1}}\ in\ z^A) : A \\
&\overset{\beta_{\multimap}}{\approx} x : A \vdash let\ \langle t^{\mathbf{1}}, z^A \rangle = \langle \star, x \rangle\ in\ (let\ \star = t^{\mathbf{1}}\ in\ z^A) : A \\
&\overset{\beta_{\otimes}}{\approx} x : A \vdash let\ \star = \star\ in\ x : A \\
&\overset{\eta_{\star}}{\approx} x : A \vdash x : A \\
&= id_A
\end{aligned}$$

and

$$
\begin{aligned}
l_A^{-1} \circ l_A &= (z : A \vdash \langle \star, z \rangle : 1 \otimes A) \circ (y : 1 \otimes A \vdash \ let\ \langle \star, x^A \rangle = y\ in\ x^A : A) \\
&= y : 1 \otimes A \vdash \ let\ z^A = \\
&\qquad\qquad let\ \langle t^{\mathbf{1}}, z^A \rangle = y\ in\ (let\ \star = t^{\mathbf{1}}\ in\ x^A)\ in\ \langle \star, z \rangle : 1 \otimes A \\
&\overset{let_1}{\approx} y : 1 \otimes A \vdash \ let\ \langle t^{\mathbf{1}}, z^A \rangle = y\ in \\
&\qquad\qquad (let\ z^A = (let\ \star = t^{\mathbf{1}}\ in\ x^A)\ in\ \langle \star, z \rangle) : 1 \otimes A \\
&\overset{let_1}{\approx} y : 1 \otimes A \vdash \ let\ \langle t^{\mathbf{1}}, z^A \rangle = y\ in \\
&\qquad\qquad (let\ \star = t^{\mathbf{1}}\ in\ (let\ z^A = x^A\ in\ \langle \star, z \rangle)) : 1 \otimes A \\
&\overset{\beta_{\multimap}}{\approx} y : 1 \otimes A \vdash \ let\ \langle t^{\mathbf{1}}, z^A \rangle = y\ in\ (let\ \star = t^{\mathbf{1}}\ in\ \langle \star, x^A \rangle) : 1 \otimes A \\
&\overset{\beta_{\multimap}}{\approx} y : 1 \otimes A \vdash \ let\ \langle t^{\mathbf{1}}, z^A \rangle = y\ in\ \langle t^{\mathbf{1}}, x^A \rangle : 1 \otimes A \\
&\overset{\eta_{\otimes}}{\approx} y : 1 \otimes A \vdash y : 1 \otimes A \\
&= id_{1 \otimes A}
\end{aligned}
$$

and analogously for r_A, r_A^{-1}. Additionally

$$
\begin{aligned}
a_{A,B,C} \circ a^{-1}_{A,B,C} = \big(&r_1 : (A \otimes B) \otimes C \vdash \ let\ \langle t^{A\otimes B}, y_1^C\rangle = r_1\ in \\
&let\ \langle w_1^A, x_1^B\rangle = t^{A\otimes B}\ in\ \langle w_1^A, \langle x_1^B, y_1^C\rangle\rangle : A \otimes (B \otimes C)\big) \circ \\
&\big(r_2 : A \otimes (B \otimes C) \vdash \ let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&let\ \langle x_2^B, y_2^C\rangle = u^{B\otimes C}\ in\ \langle\langle w_2^A, x_2^B\rangle, y_2^C\rangle : (A \otimes B) \otimes C\big) \\
= r_2 : A \otimes (B \otimes C) \vdash \ &let\ r_1 = \big(let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&(let\ \langle x_2^B, y_2^C\rangle = u^{B\otimes C}\ in\ \langle\langle w_2^A, x_2^B\rangle, y_2^C\rangle)\big)\ in \\
&\big(let\ \langle t^{A\otimes B}, y_1^C\rangle = r_1\ in \\
&(let\ \langle w_1^A, x_1^B\rangle = t^{A\otimes B}\ in\ \langle w_1^A, \langle x_1^B, y_1^C\rangle\rangle)\big) : A \otimes (B \otimes C) \\
\overset{let_1}{\approx} r_2 : A \otimes (B \otimes C) \vdash &let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&\big(let\ r_1 = (let\ \langle x_2^B, y_2^C\rangle = u^{B\otimes C}\ in\ \langle\langle w_2^A, x_2^B\rangle, y_2^C\rangle)\ in \\
&\big(let\ \langle t^{A\otimes B}, y_1^C\rangle = r_1\ in \\
&(let\ \langle w_1^A, x_1^B\rangle = t^{A\otimes B}\ in\ \langle w_1^A, \langle x_1^B, y_1^C\rangle\rangle)\big)\big) : A \otimes (B \otimes C) \\
\overset{let_1}{\approx} r_2 : A \otimes (B \otimes C) \vdash &let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&\big(let\ \langle x_2^B, y_2^C\rangle = u^{B\otimes C}\ in \\
&\big(let\ r_1 = \langle\langle w_2^A, x_2^B\rangle, y_2^C\rangle\ in \\
&\big(let\ \langle t^{A\otimes B}, y_1^C\rangle = r_1\ in \\
&(let\ \langle w_1^A, x_1^B\rangle = t^{A\otimes B}\ in\ \langle w_1^A, \langle x_1^B, y_1^C\rangle\rangle)\big)\big)\big) : A \otimes (B \otimes C) \\
\overset{\beta_{\multimap}}{\approx} r_2 : A \otimes (B \otimes C) \vdash &let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&\big(let\ \langle x_2^B, y_2^C\rangle = u^{B\otimes C}\ in \\
&\big(let\ \langle t^{A\otimes B}, y_1^C\rangle = \langle\langle w_2^A, x_2^B\rangle, y_2^C\rangle\ in \\
&(let\ \langle w_1^A, x_1^B\rangle = t^{A\otimes B}\ in\ \langle w_1^A, \langle x_1^B, y_1^C\rangle\rangle)\big)\big) : A \otimes (B \otimes C) \\
\overset{\beta_{\otimes}}{\approx} r_2 : A \otimes (B \otimes C) \vdash &let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&\big(let\ \langle x_2^B, y_2^C\rangle = u^{B\otimes C}\ in \\
&\langle w_2^A, \langle x_2^B, y_2^C\rangle\rangle\big) : A \otimes (B \otimes C) \\
\overset{\beta_{\otimes}}{\approx} r_2 : A \otimes (B \otimes C) \vdash &let\ \langle w_2^A, u^{B\otimes C}\rangle = r_2\ in \\
&\langle w_2^A, u^{B\otimes C}\rangle : A \otimes (B \otimes C) \\
\overset{\eta_{\otimes}}{\approx} r_2 : A \otimes (B \otimes C) \vdash &\ r_2 : A \otimes (B \otimes C) \\
= id_{A\otimes(B\otimes C)}\,.&
\end{aligned}
$$

$a^{-1}_{A,B,C} \circ a_{A,B,C} \approx id_{(A\otimes B)\otimes C}$ is shown in the exact same manner.

It remains to show the triangle equations $(id_A \otimes l_B) \circ a_{A,I,B} = r_A \otimes l_B$ as well as the pentagon equations $a_{A,B,C\otimes D} \circ a_{A\otimes B,C,D} = (id_A \otimes a_{B,C,D}) \circ a_{A,B\otimes C,D} \circ$

$(a_{A,B,C} \otimes id_D)$. For the triangle equations we note that

$$\begin{aligned} r_A \otimes id_B = {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & \big\langle let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ (let\ \star = t^1\ in\ y^A), w^B\big\rangle : A \otimes B \\ \overset{let^2_\otimes}{\approx} {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ \big(let\ \star = t^1\ in\ \langle y^A, w^B\rangle\big) : A \otimes B \end{aligned} \tag{3.17}$$

and thus

$$\begin{aligned} (id_A \otimes l_B) \circ a_{A,I,B} = {} & \big(z' : A \otimes (1 \otimes B) \vdash \ let\ \langle y^A, v^{1\otimes B}\rangle = z'\ in \\ & \big\langle y^A,\ let\ \langle t^1, w^B\rangle = v^{1\otimes B}\ in\ (let\ \star = t^1\ in\ w^B)\big\rangle : A \otimes B\big) \\ & \circ \big(z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ \langle y^A, \langle t^1, w^B\rangle\rangle : A \otimes (1 \otimes B)\big) \\ = {} & z : (A \otimes 1) \otimes B \vdash \ let\ z' = \big(let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & \big(let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ \langle y^A, \langle t^1, w^B\rangle\rangle\big)\ in \\ & \big(let\ \langle y^A, v^{1\otimes B}\rangle = z'\ in \\ & \big\langle y^A,\ let\ \langle t^1, w^B\rangle = v^{1\otimes B}\ in\ (let\ \star = t^1\ in\ w^B)\big\rangle\big)\big) : A \otimes B \\ \overset{let_1}{\approx} {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & \big(let\ z' = \big(let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ \langle y^A, \langle t^1, w^B\rangle\rangle\big)\ in \\ & \big(let\ \langle y^A, v^{1\otimes B}\rangle = z'\ in \\ & \big\langle y^A,\ let\ \langle t^1, w^B\rangle = v^{1\otimes B}\ in\ (let\ \star = t^1\ in\ w^B)\big\rangle\big)\big) : A \otimes B \\ \overset{\beta_{\multimap}}{\approx} {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & \big(let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in \\ & \big(let\ \langle y^A, v^{1\otimes B}\rangle = \langle y^A, \langle t^1, w^B\rangle\rangle\ in \\ & \big\langle y^A,\ let\ \langle t^1, w^B\rangle = v^{1\otimes B}\ in\ (let\ \star = t^1\ in\ w^B)\big\rangle\big)\big) : A \otimes B \\ \overset{\beta_{\multimap}}{\approx} {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & \big(let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in \\ & \big\langle y^A,\ let\ \langle t^1, w^B\rangle = \langle t^1, w^B\rangle\ in \\ & \qquad (let\ \star = t^1\ in\ w^B)\big\rangle\big) : A \otimes B \\ \overset{\beta_{\multimap}}{\approx} {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ \big\langle y^A, let\ \star = t^1\ in\ w^B\big\rangle : A \otimes B \\ \overset{let^2_\otimes}{\approx} {} & z : (A \otimes 1) \otimes B \vdash \ let\ \langle x^{A\otimes 1}, w^B\rangle = z\ in \\ & let\ \langle y^A, t^1\rangle = x^{A\otimes 1}\ in\ \big(let\ \star = t^1\ in\ \langle y^A, w^B\rangle\big) : A \otimes B \\ \overset{(3.17)}{\approx} {} & r_A \otimes id_B\,. \end{aligned}$$

To show the pentagon equation we first consider the lefthanded side.

$$
\begin{aligned}
a_{A,B,C\otimes D} \circ a_{A\otimes B,C,D} = & \langle t_2^{A\otimes B}, \langle y^C, z^D\rangle\rangle) : (A\otimes B)\otimes (C\otimes D)\big) \\
= & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \ let\ q_1^{(A\otimes B)\otimes (C\otimes D)} = \\
& \big(let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
& (let\ \langle t_2^{A\otimes B}, y^C\rangle = r^{(A\otimes B)\otimes C}\ in \\
& \langle t_2^{A\otimes B}, \langle y^C, z^D\rangle\rangle))\ in\ \big(let\ \langle t_1^{A\otimes B}, v^{C\otimes D}\rangle = q_1\ in \\
& (let\ \langle w^A, x^B\rangle = t_1^{A\otimes B}\ in \\
& \langle w^A, \langle x^B, v^{C\otimes D}\rangle\rangle))\big) : A\otimes (B\otimes (C\otimes D)) \\
\overset{let_1}{\approx} & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \ let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
& \big(let\ q_1^{(A\otimes B)\otimes (C\otimes D)} = (let\ \langle t_2^{A\otimes B}, y^C\rangle = r^{(A\otimes B)\otimes C}\ in \\
& \langle t_2^{A\otimes B}, \langle y^C, z^D\rangle\rangle)\ in\ \big(let\ \langle t_1^{A\otimes B}, v^{C\otimes D}\rangle = q_1\ in \\
& (let\ \langle w^A, x^B\rangle = t_1^{A\otimes B}\ in \\
& \langle w^A, \langle x^B, v^{C\otimes D}\rangle\rangle))\big)\big) : A\otimes (B\otimes (C\otimes D)) \\
\overset{let_1}{\approx} & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \ let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
& \big(let\ \langle t_2^{A\otimes B}, y^C\rangle = r^{(A\otimes B)\otimes C}\ in \\
& \big(let\ q_1^{(A\otimes B)\otimes (C\otimes D)} = \langle t_2^{A\otimes B}, \langle y^C, z^D\rangle\rangle\ in \\
& \big(let\ \langle t_1^{A\otimes B}, v^{C\otimes D}\rangle = q_1\ in \\
& \big(let\ \langle w^A, x^B\rangle = t_1^{A\otimes B}\ in \\
& \langle w^A, \langle x^B, v^{C\otimes D}\rangle\rangle)))\big) : A\otimes (B\otimes (C\otimes D)) \\
\overset{\beta_{\multimap}}{\approx} & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \ let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
& \big(let\ \langle t_2^{A\otimes B}, y^C\rangle = r^{(A\otimes B)\otimes C}\ in \\
& \big(let\ \langle t_1^{A\otimes B}, v^{C\otimes D}\rangle = \langle t_2^{A\otimes B}, \langle y^C, z^D\rangle\rangle\ in \\
& \big(let\ \langle w^A, x^B\rangle = t_1^{A\otimes B}\ in \\
& \langle w^A, \langle x^B, v^{C\otimes D}\rangle\rangle)))\big) : A\otimes (B\otimes (C\otimes D)) \\
\overset{\beta_{\otimes}}{\approx} & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \ let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
& \big(let\ \langle t_2^{A\otimes B}, y^C\rangle = r^{(A\otimes B)\otimes C}\ in \\
& \big(let\ \langle w^A, x^B\rangle = t_2^{A\otimes B}\ in \\
& \langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle)))\big) : A\otimes (B\otimes (C\otimes D))\,.
\end{aligned}
\tag{3.18}
$$

For the righthanded side we first look at $a_{A,B,C}\otimes id_D$ and $id_A\otimes a_{B,C,D}$ sepa-

rately.

$$\begin{aligned}
a_{A,B,C} \otimes id_D = \big(&r : (A \otimes B) \otimes C \vdash \ let\ \langle u^{A\otimes B}, y^C\rangle = r\ in \\
&let\ \langle w^A, x^B\rangle = u^{A\otimes B}\ in\ \langle w^A, \langle x^B, y^C\rangle\rangle : A \otimes (B \otimes C)\big) \\
&\otimes \big(z : D \vdash z : D\big) \\
= q : ((A \otimes B) \otimes C) \otimes D \vdash &\ let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q\ in \\
&\big\langle let\ \langle u^{A\otimes B}, y^C\rangle = r\ in\ (let\ \langle w^A, x^B\rangle = u^{A\otimes B}\ in \\
&\qquad\qquad \langle w^A, \langle x^B, y^C\rangle\rangle)\ ,\ z^D\big\rangle : (A \otimes (B \otimes C)) \otimes D \\
\overset{let^2_\otimes}{\approx} q : ((A \otimes B) \otimes C) \otimes D \vdash &\ let\ \langle r^{(A\otimes B)\otimes C}, z^D\rangle = q\ in \\
&\big(let\ \langle u^{A\otimes B}, y^C\rangle = r^{(A\otimes B)\otimes C}\ in\ \big(let\ \langle w^A, x^B\rangle = u^{A\otimes B}\ in \\
&\big\langle\langle w^A, \langle x^B, y^C\rangle\rangle, z^D\big\rangle\big) : (A \otimes (B \otimes C)) \otimes D
\end{aligned}$$

and analogously

$$\begin{aligned}
id_A \otimes a_{B,C,D} \approx q : A \otimes ((B \otimes C) \otimes D) \vdash &\ let\ \langle w^A, s^{(B\otimes C)\otimes D}\rangle = q\ in \\
&\big(let\ \langle u^{B\otimes C}, z^D\rangle = s\ in\ \big(let\ \langle x^B, y^C\rangle = u^{B\otimes C}\ in \\
&\langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle\big)\big) : A \otimes (B \otimes (C \otimes D))\,.
\end{aligned}$$

Thus we can compute, that

$$
\begin{aligned}
a_{A,B\otimes C,D}\circ(a_{A,B,C}\otimes id_D) \approx\ & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \mathit{let}\ q_3^{A\otimes(B\otimes C)} = \\
& \big(\mathit{let}\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ \mathit{in} \\
& \big(\mathit{let}\ \langle t^{A\otimes B}, y^C\rangle = r_1\ \mathit{in}\ \big(\mathit{let}\ \langle w^A, x^B\rangle = t^{A\otimes B}\ \mathit{in} \\
& \big\langle\langle w^A, \langle x^B, y^C\rangle\rangle, z^D\big\rangle\big)\big)\big)\ \mathit{in} \\
& \big(\mathit{let}\ \langle r_2^{A\otimes(B\otimes C)}, z^D\rangle = q_3\ \mathit{in} \\
& (\mathit{let}\ \langle w^A, u^{B\otimes C}\rangle = r_2^{A\otimes(B\otimes C)}\ \mathit{in} \\
& \langle w^A, \langle u^{B\otimes C}, z^D\rangle\rangle)\big) : A\otimes((B\otimes C)\otimes D) \\
\overset{let_1}{\approx}\ & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \mathit{let}\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ \mathit{in} \\
& \big(\mathit{let}\ q_3^{A\otimes(B\otimes C)} = \big(\mathit{let}\ \langle t^{A\otimes B}, y^C\rangle = r_1\ \mathit{in} \\
& \big(\mathit{let}\ \langle w^A, x^B\rangle = t^{A\otimes B}\ \mathit{in} \\
& \big\langle\langle w^A, \langle x^B, y^C\rangle\rangle, z^D\big\rangle\big)\big)\ \mathit{in} \\
& \big(\mathit{let}\ \langle r_2^{A\otimes(B\otimes C)}, z^D\rangle = q_3\ \mathit{in} \\
& (\mathit{let}\ \langle w^A, u^{B\otimes C}\rangle = r_2^{A\otimes(B\otimes C)}\ \mathit{in} \\
& \langle w^A, \langle u^{B\otimes C}, z^D\rangle\rangle)\big)\big) : A\otimes((B\otimes C)\otimes D) \\
\overset{let_1}{\approx}\ & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \mathit{let}\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ \mathit{in} \\
& \big(\mathit{let}\ \langle t^{A\otimes B}, y^C\rangle = r_1\ \mathit{in}\ \big(\mathit{let}\ \langle w^A, x^B\rangle = t^{A\otimes B}\ \mathit{in} \\
& \big(\mathit{let}\ q_3^{A\otimes(B\otimes C)} = \big\langle\langle w^A, \langle x^B, y^C\rangle\rangle, z^D\big\rangle\ \mathit{in} \\
& \big(\mathit{let}\ \langle r_2^{A\otimes(B\otimes C)}, z^D\rangle = q_3\ \mathit{in} \\
& (\mathit{let}\ \langle w^A, u^{B\otimes C}\rangle = r_2^{A\otimes(B\otimes C)}\ \mathit{in} \\
& \langle w^A, \langle u^{B\otimes C}, z^D\rangle\rangle)\big)\big)\big) : A\otimes((B\otimes C)\otimes D) \\
\overset{\beta_\multimap}{\approx}\ & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \mathit{let}\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ \mathit{in} \\
& \big(\mathit{let}\ \langle t^{A\otimes B}, y^C\rangle = r_1\ \mathit{in}\ \big(\mathit{let}\ \langle w^A, x^B\rangle = t^{A\otimes B}\ \mathit{in} \\
& \big(\mathit{let}\ \langle r_2^{A\otimes(B\otimes C)}, z^D\rangle = \big\langle\langle w^A, \langle x^B, y^C\rangle\rangle, z^D\big\rangle\ \mathit{in} \\
& (\mathit{let}\ \langle w^A, u^{B\otimes C}\rangle = r_2^{A\otimes(B\otimes C)}\ \mathit{in} \\
& \langle w^A, \langle u^{B\otimes C}, z^D\rangle\rangle)\big)\big)\big) : A\otimes((B\otimes C)\otimes D) \\
\overset{\beta_\otimes}{\approx}\ & q_2 : ((A\otimes B)\otimes C)\otimes D \vdash \mathit{let}\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ \mathit{in} \\
& \big(\mathit{let}\ \langle t^{A\otimes B}, y^C\rangle = r_1\ \mathit{in}\ \big(\mathit{let}\ \langle w^A, x^B\rangle = t^{A\otimes B}\ \mathit{in} \\
& \langle w^A, \langle\langle x^B, y^C\rangle, z^D\rangle\rangle\big)\big) : A\otimes((B\otimes C)\otimes D)\,.
\end{aligned}
$$

Overall we then have

$$
\begin{aligned}
&(id_A \otimes a_{B,C,D}) \circ a_{A,B\otimes C,D} \circ (a_{A,B,C} \otimes id_D) \approx \\
&\qquad q_2 : ((A \otimes B) \otimes C) \otimes D \vdash \ let\ q_3^{A\otimes((B\otimes C)\otimes D)} = \\
&\qquad \big(let\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
&\qquad \big(let\ \langle t^{A\otimes B}, y^C\rangle = r_1\ in\ \big(let\ \langle w^A, x^B\rangle = t^{A\otimes B}\ in \\
&\qquad \langle w^A, \langle\langle x^B, y^C\rangle, z^D\rangle\rangle\big)\big)\big)\ in \\
&\qquad \big(let\ \langle w^A, s^{(B\otimes C)\otimes D}\rangle = q_3\ in \\
&\qquad \big(let\ \langle u^{B\otimes C}, z^D\rangle = s\ in\ \big(let\ \langle x^B, y^C\rangle = u^{B\otimes C}\ in \\
&\qquad \langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle\big)\big)\big) : A \otimes (B \otimes (C \otimes D)) \\
&\overset{let_1}{\approx} q_2 : ((A \otimes B) \otimes C) \otimes D \vdash \ let\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
&\qquad \big(let\ q_3^{A\otimes((B\otimes C)\otimes D)} = \\
&\qquad \big(let\ \langle t^{A\otimes B}, y^C\rangle = r_1\ in\ \big(let\ \langle w^A, x^B\rangle = t^{A\otimes B}\ in \\
&\qquad \langle w^A, \langle\langle x^B, y^C\rangle, z^D\rangle\rangle\big)\big)\ in \\
&\qquad \big(let\ \langle w^A, s^{(B\otimes C)\otimes D}\rangle = q_3\ in \\
&\qquad \big(let\ \langle u^{B\otimes C}, z^D\rangle = s\ in\ \big(let\ \langle x^B, y^C\rangle = u^{B\otimes C}\ in \\
&\qquad \langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle\big)\big)\big)\big) : A \otimes (B \otimes (C \otimes D)) \\
&\overset{let_1}{\approx} q_2 : ((A \otimes B) \otimes C) \otimes D \vdash \ let\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
&\qquad \big(let\ \langle t^{A\otimes B}, y^C\rangle = r_1\ in \\
&\qquad \big(let\ \langle w^A, x^B\rangle = t^{A\otimes B}\ in \\
&\qquad \big(let\ q_3^{A\otimes((B\otimes C)\otimes D)} = \langle w^A, \langle\langle x^B, y^C\rangle, z^D\rangle\rangle\ in \\
&\qquad \big(let\ \langle w^A, s^{(B\otimes C)\otimes D}\rangle = q_3\ in \\
&\qquad \big(let\ \langle u^{B\otimes C}, z^D\rangle = s\ in\ \big(let\ \langle x^B, y^C\rangle = u^{B\otimes C}\ in \\
&\qquad \langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle\big)\big)\big)\big)\big)\big) : A \otimes (B \otimes (C \otimes D)) \\
&\overset{\beta_{\multimap}}{\approx} q_2 : ((A \otimes B) \otimes C) \otimes D \vdash \ let\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
&\qquad \big(let\ \langle t^{A\otimes B}, y^C\rangle = r_1\ in \\
&\qquad \big(let\ \langle w^A, x^B\rangle = t^{A\otimes B}\ in \\
&\qquad \big(let\ \langle w^A, s^{(B\otimes C)\otimes D}\rangle = \langle w^A, \langle\langle x^B, y^C\rangle, z^D\rangle\rangle\ in \\
&\qquad \big(let\ \langle u^{B\otimes C}, z^D\rangle = s\ in\ \big(let\ \langle x^B, y^C\rangle = u^{B\otimes C}\ in \\
&\qquad \langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle\big)\big)\big)\big)\big)\big) : A \otimes (B \otimes (C \otimes D)) \\
&\overset{\beta_{\otimes}}{\approx} q_2 : ((A \otimes B) \otimes C) \otimes D \vdash \ let\ \langle r_1^{(A\otimes B)\otimes C}, z^D\rangle = q_2\ in \\
&\qquad \big(let\ \langle t^{A\otimes B}, y^C\rangle = r_1\ in \\
&\qquad \big(let\ \langle w^A, x^B\rangle = t^{A\otimes B}\ in \\
&\qquad \langle w^A, \langle x^B, \langle y^C, z^D\rangle\rangle\rangle\big)\big) : A \otimes (B \otimes (C \otimes D)) \\
&\overset{(3.18)}{\approx} a_{A,B,C\otimes D} \circ a_{A\otimes B,C,D}\,.
\end{aligned}
$$

This shows, that the pentagon equation holds, proving that $(\mathbf{Val}, \otimes)$ is monoidal.

Step 3: $(\mathbf{Val}, \otimes)$ admits a symmetric braiding.
We define the braiding $b_{A,B} : A \otimes B \longrightarrow B \otimes A$ as follows:

$$b_{A,B} := \ z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in\ \langle y, x\rangle : B \otimes A\,.$$

We first show, that it is symmetric, i.e. $b_{A,B}^{-1} = b_{B,A}$.

$$\begin{aligned}
b_{B,A} \circ b_{A,B} &= \big(z' : B \otimes A \vdash \ let\ \langle y', x'\rangle = z'\ in\ \langle x', y'\rangle : A \otimes B\big) \circ \\
&\quad \big(z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in\ \langle y, x\rangle : B \otimes A\big) \\
&= z : A \otimes B \vdash \ let\ z' = \big(let\ \langle x, y\rangle = z\ in \\
&\quad \langle y, x\rangle\big)\ in\ \big(let\ \langle y', x'\rangle = z'\ in\ \langle x', y'\rangle\big) : A \otimes B \\
&\overset{let_1}{\approx} z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\
&\quad \big(let\ z' = \langle y, x\rangle\ in\ \big(let\ \langle y', x'\rangle = z'\ in\ \langle x', y'\rangle\big)\big) : A \otimes B \\
&\overset{\beta_{\multimap}}{\approx} z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in \\
&\quad \big(let\ \langle y', x'\rangle = \langle y, x\rangle\ in\ \langle x', y'\rangle\big) : A \otimes B \\
&\overset{\beta_{\multimap}}{\approx} z : A \otimes B \vdash \ let\ \langle x, y\rangle = z\ in\ \langle x, y\rangle : A \otimes B \\
&\overset{\eta_{\otimes}}{\approx} z : A \otimes B \vdash z : A \otimes B \\
&= id_{A \otimes B}
\end{aligned}$$

The braiding also needs to satisfy the hexagon equations:

$$\begin{aligned}
b_{A,B\otimes C} &= a^{-1}_{B,C,A} \circ (id_B \otimes b_{A,C}) \circ a_{B,A,C} \circ (b_{A,B} \otimes id_C) \circ a^{-1}_{A,B,C} \\
b_{A\otimes B,C} &= a_{C,A,B} \circ (b_{A,C} \otimes id_B) \circ a^{-1}_{A,C,B} \circ (id_A \otimes b_{B,C}) \circ a_{A,B,C}
\end{aligned}$$

The calculations are again straightforward but very lenghty and will thus be skipped. □

Lemma 3.44. $(\mathbf{cVal}, \otimes, 1)$ *is cartesian.*

Proof. We start by showing that $\mathsf{1}$ is terminal. For that let A be any type, then we can simply construct

$$\nabla_A := \ x : !A \vdash \star : \mathsf{1}\,.$$

For the product we need to construct projections $\pi^1_{A,B}$, $\pi^2_{A,B}$ for any types A, B. We define them to be

$$\begin{aligned}
\pi^1_{A,B} &:= \ w : !(A \otimes B) \vdash \ let\ \langle x, y\rangle = w\ in\ x : !A \\
\pi^2_{A,B} &:= \ w : !(A \otimes B) \vdash \ let\ \langle x, y\rangle = w\ in\ y : !B\,.
\end{aligned}$$

Now for any pair of morphisms $f : C \longrightarrow A, g : C \longrightarrow B$, say $f = \ z : !C \vdash U : !A$, $g = \ z : !C \vdash V : !B$ we set

$$\langle f, g\rangle := \ z : !C \vdash \langle U, V\rangle : !A \otimes !B\,.$$

Clearly $\langle f, g\rangle : C \longrightarrow A \otimes B$ and

$$\begin{aligned}
\pi^1_{A,B} \circ \langle f, g\rangle &= \big(w : !(A \otimes B) \vdash \ let\ \langle x, y\rangle = w\ in\ x : !A\big) \circ \\
&\quad \big(z : !C \vdash \langle U, V\rangle : !A \otimes !B\big) \\
&= z : !C \vdash let\ w = \langle U, V\rangle\ in\ \big(let\ \langle x, y\rangle = w\ in\ x\big) : !A \\
&\overset{\beta_{\multimap}}{\approx} z : !C \vdash let\ \langle x, y\rangle = \langle U, V\rangle\ in\ x : !A \\
&\overset{\beta_{\otimes}}{\approx} z : !C \vdash U : !A \\
&= f\,.
\end{aligned}$$

Analogously one shows $\pi^2_{A,B} \circ \langle f, g\rangle = g$. It remains to show, that for all $h : C \longrightarrow A \otimes B$ we have $\langle \pi^1_{A,B} \circ h, \pi^2_{A,B} \circ h\rangle = h$. So let $h = \ z : !C \vdash \langle U, V\rangle : !A \otimes !B$, then

$$\begin{aligned}
\pi^1_{A,B} \circ h = z : !C \vdash let\ w = \langle U, V\rangle\ in\ \big(let\ \langle x, y\rangle = w\ in\ x\big) &: !A \\
\overset{\beta_{\multimap}}{\approx}\ z : !C \vdash let\ \langle x, y\rangle = \langle U, V\rangle\ in\ x &: !A \\
\overset{\beta_{\otimes}}{\approx}\ z : !C \vdash U &: !A
\end{aligned}$$

and analogously $\pi^2_{A,B} \circ h \approx \ z : !C \vdash V : !B$, thus

$$\langle \pi^1_{A,B} \circ h, \pi^2_{A,B} \circ h\rangle = \ z : !C \vdash \langle U, V\rangle : !A \otimes !B \ = h\,.$$

□

3.2.3 Semantics of the Function Type

To analyze the semantics of the function type, we have to take a look at the differences between then notions of a *value* and a *computation*. We have defined values such that they can be interpreted as a result of a computation (cmp. section 3.1.4). Computations on the other hand are arbitrary terms. If quantum computers were deterministic, we could easily identify each computation with its resulting value, giving us a surjective embedding of computations into values. But since that is not the case, a computation will rather result in a probability distribution over values, so this identification can not work. We have however at the end of section 2.2.5 found a bijection between propositions A and propositions of kind $1 \multimap A$. We can make use of that, since terms of type $1 \multimap A$ are always values. Translating the derivations (2.26), (2.27) into the typing system we can derive from $\Gamma \vdash M : A$ the judgement $\Gamma \vdash \lambda\star.M : 1 \multimap A$ and the other way around we can derive from $\Gamma \vdash V : 1 \multimap A$ the judgement $\Gamma \vdash V\star : A$. We can thus identify with each computation $M : A$ the value of the function $\lambda\star.M : 1 \multimap A$ which executes the computation, and with each function of the form $V : 1 \multimap A$ the computation $V\star : A$ it executes. These maps are mutually inverse up to equivalence.

These ideas can be incorporated into our categories using the concepts of a *monad* and related structures, which we introduce now [17].

Definition 3.45 (Monad)**.** A **monad** over a category $\mathcal{C}$ consists of

- A functor $T : \mathcal{C} \longrightarrow \mathcal{C}$

- A natural transformations $\eta : id_{\mathcal{C}} \Longrightarrow T$ called the **unit**
- A natural transformation $\mu : T^2 \Longrightarrow T$ called the **multiplication**

such that the following diagrams commute:

$$\begin{array}{ccc} T^3A & \xrightarrow{T\mu_A} & T^2A \\ {\scriptstyle \mu_{TA}}\downarrow & & \downarrow{\scriptstyle \mu_A} \\ T^2A & \xrightarrow{\mu_A} & TA \end{array} \qquad \begin{array}{ccccc} TA & \xrightarrow{\eta_{TA}} & T^2A & \xleftarrow{T\eta_A} & TA \\ & {\scriptstyle id_{TA}}\searrow & \downarrow{\scriptstyle \mu_A} & \swarrow{\scriptstyle id_{TA}} & \\ & & TA & & \end{array}$$

In the case of a monoidal category $(\mathcal{C}, \otimes, I, a, l, r)$ we call a monad (T, η, μ) **strong** if there is a natural transformation $t_{A,B} : A \otimes TB \longrightarrow T(A \otimes B)$, called the **tensorial strength**, such that the following diagrams commute:

$$\begin{array}{ccc} 1 \otimes TA & \xrightarrow{t_{1,A}} & T(1 \otimes A) \\ & {\scriptstyle l_{TA}}\searrow & \downarrow{\scriptstyle Tl_A} \\ & & TA \end{array} \qquad \begin{array}{ccc} A \otimes B & \xrightarrow{id_A \otimes \eta_B} & A \otimes TB \\ & {\scriptstyle \eta_{A\otimes B}}\searrow & \downarrow{\scriptstyle t_{A,B}} \\ & & T(A \otimes B) \end{array}$$

$$\begin{array}{ccc} (A \otimes B) \otimes TC & \xrightarrow{t_{A\otimes B,C}} & T((A \otimes B) \otimes C) \\ \downarrow{\scriptstyle a_{A,B,TC}} & & \downarrow{\scriptstyle Tl_A} \\ A \otimes (B \otimes TC) & \xrightarrow{id_A \otimes t_{B,C}} A \otimes T(B \otimes C) \xrightarrow{t_{A,B\otimes C}} & T(A \otimes (B \otimes C)) \end{array}$$

$$\begin{array}{ccccc} A \otimes TTB & \xrightarrow{t_{A,TB}} & T(A \otimes TB) & \xrightarrow{Tt_{A,B}} & TT(A \otimes B) \\ \downarrow{\scriptstyle id_A \otimes \mu_B} & & & & \downarrow{\scriptstyle \mu_{A\otimes B}} \\ A \otimes TB & & \xrightarrow{\qquad t_{A,B} \qquad} & & T(A \otimes B) \end{array}$$

Definition 3.46 (Kleisli category)**.** Let $\mathcal{C}$ be a category and (T, η, μ) a monad over $\mathcal{C}$. Then the **Kleisli category** $\mathcal{C}_T$ is the category with the following properties

- the objects of $\mathcal{C}_T$ are the objects of $\mathcal{C}$
- an arrow $f : A \longrightarrow B$ in $\mathcal{C}_T$ is an arrow $f : A \longrightarrow TB$ of $\mathcal{C}$
- the identity id_A in $\mathcal{C}_T$ is given by the arrow η_A of $\mathcal{C}$
- for arrows $f : A \longrightarrow B$, $g : B \longrightarrow C$ in $\mathcal{C}_T$ $g \circ f : A \longrightarrow C$ is defined by $\mu_C \circ Tg \circ f : A \longrightarrow TC$

Lemma 3.47. *Let A, B be objects of* **Val** *and $f : A \longrightarrow B$ an arrow, say $f = \ x : A \vdash V : B$. We define the endofunctor $T :$* **Val** $\longrightarrow$ **Val** *through*

$$\begin{aligned} TA &= 1 \multimap A \\ Tf &= \ y : 1 \multimap A \vdash \lambda\star.let\ x = (y\star)\ in\ V : 1 \multimap B\,. \end{aligned}$$

Furthermore we define the maps

$$\begin{aligned}\eta_A = &\quad x : A \vdash \lambda\star.x : 1 \multimap A\\ \mu_A = &\quad x : 1 \multimap (1 \multimap A) \vdash \lambda\star.(x\star)\star : 1 \multimap A\\ t_{A,B} = &\quad z : A \otimes (1 \multimap B) \vdash let\ \langle x, y\rangle = z\ in\ \lambda\star.\langle x, y\star\rangle : 1 \multimap (A \otimes B)\,.\end{aligned}$$

Then (T, η, μ, t) is a strong monad over **Val**.

Proof. The derivations of the judgements Tf and $t_{A,B}$ follow along the lines of derivations (2.28) and (2.29) respectively. Showing the commutation of the diagrams is straightfoward. The calculations can be found in [25] (lemma 9.3.7 there). □

Corollary 3.48. *Let T be the strong monad over* **Val** *as defined in lemma 3.47. Then* **Comp** *is the corresponding Kleisli category, i.e.*

$$\mathbf{Comp} = \mathbf{Val}_T\,. \tag{3.19}$$

Proof. Follows immediately from lemma 3.47 and the definition of a Kleisli category. □

We have thus encoded the relationship between values and computations categorically. But the implication $\multimap$ provides even more structure.

Definition 3.49 (Kleisli exponentials)**.** Let $(\mathcal{C}, \otimes, 1)$ be a symmetric monoidal category with a strong monad (T, η, μ) and $\mathcal{C}_T$ the corresponding Kleisli category. Then $\mathcal{C}$ is said to have **T-exponentials**, or **Kleisli exponentials**, if it is equipped with a bifunctor $\multimap\colon \mathcal{C}^{op} \times \mathcal{C} \longrightarrow \mathcal{C}$ and a natural isomorphism that assigns to each objects A, B, C the bijection

$$\Phi_{A,B,C}\colon \hom_{\mathcal{C}}(A, B \multimap C) \longrightarrow \hom_{\mathcal{C}_T}(A \otimes B, C) = \hom_{\mathcal{C}}(A \otimes B, TC)\,.$$

Lemma 3.50. *Let $f : A_1 \longrightarrow A_2$, $g : B_1 \longrightarrow B_2$ be arrows in* **Val**, *say $f = \ x : A_1 \vdash V : A_2$, $\quad g = \ y : B_1 \vdash W : B_2$. We define $\multimap\colon \mathbf{Val}^{op} \times \mathbf{Val} \longrightarrow \mathbf{Val}$ to be the bifunctor defined on arrows by*

$$\begin{aligned}A \multimap g = &\quad m : A \multimap B_1 \vdash \lambda z.(let\ y = mz\ in\ W) : A \multimap B_2\,,\\ f \multimap B = &\quad m : A_2 \multimap B \vdash \lambda x.(mV) : A_1 \multimap B\,.\end{aligned}$$

Additionally we define the map $\Phi_{A,B,C} : \hom_{\mathbf{Val}}(A, B \multimap C) \longrightarrow \hom_{\mathbf{Val}}(A \otimes B, 1 \multimap C)$ on an arrow $f = x : A \vdash V : B \multimap C$ by

$$\Phi_{A,B,C}(f) = \ p : A \otimes B \vdash \lambda\star.(let\ \langle x, y\rangle = t\ in\ Vy) : 1 \multimap C\,.$$

Then for $(\mathbf{Val}, \otimes, 1)$ with the monad (T, η, μ) as defined in lemma 3.47, $\multimap$ and Φ yield T-exponentials.

Proof. The validity of the judgements $A \multimap g$ and $f \multimap B$ follows from the judgements f and g. It is derived along the lines of derivations (2.24). The judgement $\Phi_{A,B,C}(f)$ can be derived from f as done in (2.30). □

3.2.4 Semantics of the Bang Operator "!"

To describe the operator !, we first need to introduce additional categorical notions.

Definition 3.51 (Comonad)**.** Let $\mathcal{C}$ be a category. Then a **comonad** (L, ϵ, δ) consists of an endofunctor $L : \mathcal{C} \longrightarrow \mathcal{C}$, and natural transformations $\epsilon : L \Longrightarrow id_{\mathcal{C}}$, $\delta : L \Longrightarrow L^2$ which make the following diagrams commute:

$$\begin{array}{ccc} L^3A & \xleftarrow{L\delta_A} & L^2A \\ {\scriptstyle \delta_{LA}}\uparrow & & \uparrow{\scriptstyle \delta_A} \\ L^2A & \xleftarrow[\delta_A]{} & LA \end{array} \qquad \begin{array}{ccccc} LA & \xleftarrow{\epsilon_{LA}} & L^2A & \xrightarrow{L\epsilon_A} & LA \\ & {\scriptstyle id_{LA}}\nwarrow & {\scriptstyle \delta_A}\uparrow & \nearrow{\scriptstyle id_{LA}} & \\ & & LA & & \end{array}$$

Additionally we call the comonad **idempotent** if δ is an isomorphism.

Lemma 3.52. *We can extend the definition of* ! *to arrows* $f = \ x : A \vdash V : B$ *by defining*

$$!f = \ x : !A \vdash V : !B\,. \tag{3.20}$$

Let additionally ϵ, δ *be defined by*

$$\epsilon_A = \ x : !A \vdash x : A \qquad\qquad \delta_A = \ x : !A \vdash x : !^2A\,. \tag{3.21}$$

Then $(!, \epsilon, \delta)$ *is an idempotent comonad in* **Val**.

Proof. The judgement $!f$ follows from f by applying the $(L!)$ rule and then the $(R!)$ rule (see lemma 3.28). The validity of ϵ_A follows from applying the $(L!)$ rule to the identity $id_A = \ x : A \vdash x : A$ and the validity of δ_A from applying the $(R!)$ rule twice to ϵ_A. The necessary equations follow from the equivalence relation by direct calculations (see lemma 9.3.13 in [25]). □

Let us now categorify how the bang operator is compatible with the monoidal structure.

Definition 3.53 (Monoidal Comonad)**.** Let $(\mathcal{C}, \otimes, 1)$ be a symmetric monoidal category. A **monoidal comonad** on $\mathcal{C}$ is a comonad (L, δ, ϵ) equipped with natural transformations with components $d_{A,B} : LA \otimes LB \longrightarrow L(A \otimes B)$, $d_1 : 1 \longrightarrow L1$ making (L, d) a lax symmetric monoidal functor, such that the following diagrams commute:

$$\begin{array}{ccc} LA \otimes LB & \xrightarrow{d_{A,B}} & L(A \otimes B) \\ & {\scriptstyle \epsilon_A \otimes \epsilon_B}\searrow & \downarrow{\scriptstyle \epsilon_{A\otimes B}} \\ & & A \otimes B \end{array} \qquad \begin{array}{ccc} 1 & \xrightarrow{d_1} & L1 \\ & {\scriptstyle id_1}\searrow & \downarrow{\scriptstyle \epsilon_1} \\ & & 1 \end{array}$$

$$\begin{array}{ccccc} LA \otimes LB & & \xrightarrow{\qquad\qquad d_{A,B} \qquad\qquad} & & L(A \otimes B) \\ {\scriptstyle \delta_A \otimes \delta_A}\downarrow & & & & \downarrow{\scriptstyle \delta_{A\otimes B}} \\ L^2A \otimes L^2B & \xrightarrow{d_{LA,LB}} & L(LA \otimes LB) & \xrightarrow{Ld_{A,B}} & L^2(A \otimes B) \end{array} \qquad \begin{array}{ccc} 1 & \xrightarrow{d_1} & L1 \\ {\scriptstyle d_1}\downarrow & & \downarrow{\scriptstyle \delta_1} \\ L1 & \xrightarrow{Ld_1} & L^21\,. \end{array}$$

Definition 3.54 (Linear Exponential Comonad)**.** Let $(\mathcal{C}, \otimes, 1, a, l, r, b)$ be a symmetric monoidal category. Then a monoidal comonad $(L, \delta, \epsilon, d, d)$ over $\mathcal{C}$ is a **linear exponential comonad** if the following hold true:

- for each object A, LA is a commutative comonoid $(LA, \Delta_A, \nabla_A^L)$, where $\Delta_A : LA \longrightarrow LA \otimes LA$, $\nabla_A^L : LA \longrightarrow 1$
- Δ and ∇^L are monoidal natural transformations i.e. for each pair of objects A, B the following commute

$$\begin{array}{ccc} LA \otimes LB & \xrightarrow{\Delta_A \otimes \Delta_B} & (LA \otimes LA) \otimes (LB \otimes LB) \\ \downarrow{\scriptstyle d_{A,B}} & & \downarrow{\scriptstyle sw_{LA,LB}} \\ & & (LA \otimes LB) \otimes (LA \otimes LB) \\ & & \downarrow{\scriptstyle d_{A,B} \otimes d_{A,B}} \\ L(A \otimes B) & \xrightarrow{\Delta_{A \otimes B}} & L(A \otimes B) \otimes L(A \otimes B) \end{array}$$

$$\begin{array}{ccc} LA \otimes LB & \xrightarrow{\nabla_A^L \otimes \nabla_B^L} & 1 \otimes 1 \\ \downarrow{\scriptstyle d_{A,B}} & & \downarrow{\scriptstyle l_1} \\ L(A \otimes B) & \xrightarrow{\nabla_{A \otimes B}^L} & L(A \otimes B) \otimes L(A \otimes B) \end{array} \qquad \begin{array}{ccc} 1 & \xleftarrow{l_1} & 1 \otimes 1 \\ \downarrow{\scriptstyle d_1} & & \downarrow{\scriptstyle d_1 \otimes d_1} \\ L1 & \xrightarrow{\Delta_1} & L1 \otimes L1 \end{array}$$

$$\begin{array}{ccc} 1 & \xrightarrow{id_1} & 1 \\ & {\scriptstyle d_1} \searrow \quad \nearrow {\scriptstyle \nabla_1^L} & \\ & L1 & \end{array}$$

where sw is the canonical map that switches the two middle objects, i.e.

$$sw_{A,B} = a^{-1}_{A,B,A\otimes B} \circ (id_A \otimes a_{B,A,B}) \circ (id_A \otimes (b_{A,B} \otimes id_B)) \\ \circ (id_A \otimes a^{-1}_{A,B,B}) \circ a_{A,A,B\otimes B}\,.$$

- The maps

$$\Delta_A : (LA, \delta_A) \longrightarrow (LA \otimes LA, (\delta_A \otimes \delta_A); d_A)$$
$$\nabla_A^L : (LA, \delta_A) \longrightarrow (1, d_1)$$

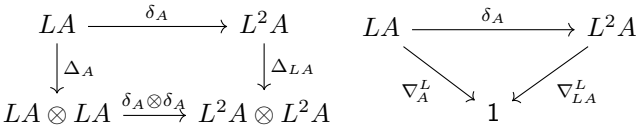
are L-coalgebra morphisms, i.e. the following commute:

$$\begin{array}{ccc} LA & \xrightarrow{\Delta_A} & LA \otimes LA \\ \downarrow{\scriptstyle \delta_A} & & \downarrow{\scriptstyle \delta_A \otimes \delta_A} \\ L^2A & \xrightarrow{L\Delta_A} & L^2A \otimes L^2A \\ & & \downarrow{\scriptstyle d_{LA,LA}} \\ & & L(LA \otimes LA) \end{array} \qquad \begin{array}{ccc} LA & \xrightarrow{\nabla_A^L} & 1 \\ \downarrow{\scriptstyle \delta_A} & & \downarrow{\scriptstyle d_1} \\ L^2A & \xrightarrow[L\nabla_A^L]{} & L1 \end{array}$$

- For each A δ_A is a comonoid morphism

$$\delta_A : (LA, \Delta_A, \nabla_A^L) \longrightarrow (L^2A, \Delta_{LA}, \nabla_{LA}^L)$$

i.e. the following commute:



Lemma 3.55. *In* **Val** *with the maps*

$$\begin{aligned} d_{A,B} &:= && z : !A \otimes !B \vdash \mathit{let}\ \langle x, y\rangle = z\ \mathit{in}\ \langle x, y\rangle : !(A \otimes B) \\ d_1 &:= && z : 1 \vdash \mathit{let}\ \star = z\ \mathit{in}\ \star : !1 \\ \Delta_A &:= && x : !A \vdash \langle x, x\rangle : !A \otimes !A \\ \nabla^!_A &:= && x : !A \vdash \star : 1\,. \end{aligned}$$

the comonad $(!, \delta, \epsilon)$ *is an idempotent linear exponential comonad.*

Proof. The judgement $d_{A,B}$ can be derived as shown in (2.25). The derivation of d_1 is

$$\dfrac{\dfrac{}{z : 1 \vdash z : 1} \qquad \dfrac{}{\vdash \star : !1}\,(\nabla^!)}{z : 1 \vdash \mathit{let}\ \star = z\ \mathit{in}\ \star : !^n 1}\,(1E)$$

The $\nabla^!_A$ judgement is just a special case of the $(\nabla^!)$ rule and Δ_A can be derived as shown in (2.21). □

An analogous sequent to d_1 can be derived in linear logic, too, namely $1 \vdash !1$. However the derivation in linear logic differs from the one here, due to our choices to use the $(\nabla^!)$ rule in our typing system and to not have an explicit $(!R)$ rule. A notable difference from linear logic without terms, is that here the arrow $d_{A,B}$ is an isomorphism. Indeed we can derive

$$\dfrac{\dfrac{}{w : !(A \otimes B) \vdash w : !(A \otimes B)} \qquad \dfrac{\dfrac{}{x : !A \vdash x : !A} \qquad \dfrac{}{y : !B \vdash y : !B}\,(\otimes I)}{x : !A, y : !B \vdash \langle x, y\rangle : !A \otimes !B}\,(\otimes E)}{w : !(A \otimes B) \vdash \mathit{let}\ \langle x, y\rangle = w\ \mathit{in}\ \langle x, y\rangle : !A \otimes !B}$$

Ultimately this difference stems from the way we chose to incorporate the !-operator into the $(\otimes E)$ rule. There we allowed for a reusable pair of data to be substituted for a pair of individually reusable data. Since a duplicable pair of data necessarily belongs to the classical control of the system and not to the quantum part, both components are classical, too. Thus allowing each of the components to be reusable individually is appropriate. The same is generally not true in linear logic, where we allow duplicable pairs of individually non-duplicable propositions.

Lemma 3.56. cVal *is the co-Kleisli category of the comonad* $(!, \delta, \epsilon)$.

□

3.2.5 Combined Structure

The combined structure looks as follows.

Definition 3.57 (Linear Category for Duplication)**.** A **linear category for duplication** is a category $\mathcal{C}$ with

- a symmetric monoidal structure $(\otimes, \mathsf{1}, a, l, r, b)$
- an idempotent, strongly monoidal, linear exponential comonad $(L, \delta, \epsilon, d, d, \Delta, \nabla)$
- a strong monad (T, μ, η, t)
- a Kleisli exponential $\multimap$, i.e. a bifunctor $\multimap\colon \mathcal{C}^{op} \times \mathcal{C} \longrightarrow \mathcal{C}$ and a natural transformation with components

$$\Phi_{A,B,C} : \hom(A \otimes B, TC) \longrightarrow \hom(A, B \multimap C)\,.$$

We call $\mathcal{C}$ **weak** if $\mathsf{1}$ is a terminal object.

Theorem 3.58. Val *is a (non weak) linear category for duplication*

Proof. This follows immediately from the previous lemmata. □

Weakness is obtained when adding the weakening rule to the system i.e. when founding the typing system on affine logic [22]. For a weak linear category of dublication a concrete model has been found [16]. For a non-weak linear category of dublication, finding a concrete model is to the author's best knowledge still an open problem.

Chapter 4

Conclusions and Future Work

4.1 Current and Future Research

4.1.1 Parallel-Non-Parallel Logic

We have seen in chapter 2, how a fragment of linear logic forms the internal logic of closed symmetric monoidal categories. However we had also seen that the category **Hilb** admits more structure, it is a dagger compact closed category, a status linear logic cannot attain. This begs the question of what an internal logic of a dagger compact closed category would look like. Here we quickly sketch a possible answer.

What is missing from linear logic is a notion of a (strong) dual. The obvious candidate $A^* := A \multimap 1$ is only a weak dual; it admits evaluation $A \otimes A^* \vdash 1$, but not coevaluation. Since $\parr$ has the structure of a monoidal product, too, we have the second option of $A^* := A \multimap \bot$. Here it is reversed, we have coevaluation $\bot \vdash A^* \parr A$, not however evaluation. We thus seek to merge the connectives $\otimes$ and $\parr$ and their units 1 and $\bot$. While that might seem odd at first glance, there is an interpretation to this. We mentioned in section 2.2.1 how the fundamental difference between conjunction and disjunction is that in the former you have the choice, while in the latter the choice is made for you. Considering again the example of buying goods from a vendor, we see that, what is a disjunction to us, is a conjunction from the perspective of the vendor and vice versa. Both perspectives would be merged when taking a third persons point of view. To a third person the situation is symmetrical, they not care who gets to choose and who doesn't, so to them there is no distinction between conjunction and disjunction. It does however still make a difference wether goods are available in parallel or not, i.e. wether it is a multiplicative connective or an additive one. We thus name this logic *parallel-non-parallel logic*.

Having sketched an interpretation, we define the logic, in particular the new connective $\shortparallel$ and its unit $\varnothing$. We read $A \shortparallel B$ as *A parallel to B*. The rules are as follows.

Axioms

$$\frac{}{A \vdash A}\ (i) \qquad \frac{}{\vdash \varnothing}\ (R\varnothing) \qquad \frac{}{\varnothing \vdash}\ (L\varnothing)$$

Logical Rules

$$\frac{A, \Gamma_1 \vdash \Gamma_2 \qquad B, \Delta_1 \vdash \Delta_2}{A \parr B, \Gamma_1, \Delta_1 \vdash \Gamma_2, \Delta_2}\ (L\parr) \qquad \frac{\Gamma_1 \vdash \Gamma_2, A \qquad \Delta_1 \vdash \Delta_2, B}{\Gamma_1, \Delta_1 \vdash \Gamma_2, \Delta_2, A \parr B}\ (R\parr)$$

$$\frac{\Gamma \vdash A \qquad B, \Delta_1 \vdash \Delta_2}{A \multimap B, \Gamma, \Delta_1 \vdash \Delta_2}\ (L\multimap) \qquad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\ (R\multimap)$$

Structural Rules

$$\frac{\Gamma_1 \vdash \Gamma_2 \qquad \Delta_1 \vdash \Delta_2}{\Gamma_1, \Delta_1 \vdash \Gamma_2, \Delta_2}\ (mix) \qquad \frac{\Gamma \vdash A \qquad A, \Delta_1 \vdash \Delta_2}{\Gamma, \Delta_1 \vdash \Delta_2}\ (cut)$$

The rule $(L\parr)$ coincides with the left rule for $\parr$, while $(R\parr)$ is just $(R\otimes)$; similarly for the unit $\varnothing$. The key to the system is the so called *mixing rule*. Using (mix) it can be shown that $\parr$ satisfies the respective other rules of $\parr$ and $\otimes$, too. Since implication is defined as in linear logic and $\parr$ satisfies both $(L\otimes)$ and $(R\otimes)$, we can safely assume that this is a closed symmetric monoidal theory (cmp. section 2.2.4). Moreover it can be proven that for $A^* := A \multimap \varnothing$ we have

$$A^* \otimes A \simeq \varnothing$$

making A^* a strong dual. We also have $A^* \parr B \simeq A \multimap B$. The additive connectives would merge into a biproduct, which is both a product and a coproduct.

A system of this kind has already been developed by R. Duncan, S. Abramsky and B. Coecke [11, 3, 2, 10] under the name *quantum logic*. It has however to the author's best knowledge never been translated into a term language in the manner that we have shown in this thesis. Such a term language would allow us to write down quantum processes as lambda terms, which might be of theoretical interest in quantum physics.

4.1.2 Dependent Linear Type Theory

An ongoing research area is the development of a dependent version of linear type theory. In such a theory the types should be divided into a linear and a classical part as, usually in linear type theory. Dependencies on linear types should be forbidden, however both linear and classical types are allowed to depend on classical types. One possible version would be a linear homotopy type theory. A discussion of the various proposed approaches can be found in [20] (section 1.7 there). It is suggested that a dependent linear type theory is the most appropriate choice for the foundations of quantum programming languages [12].

4.2 Summary and Conclusions

In this thesis we have shown how the mathematical theory behind quantum mechanics can be abstracted using categorical methods. More concretely we showed, that it forms a *dagger compact closed category*. Subsequently we analyzed a fragment of linear logic and realized that it forms a sufficiently similar structure, namely a *closed symmetric theory*. This motivated us to build the lambda calculus for quantum computation on linear logic. We went on to define a term language for the lambda calculus and introduced a typing system based on linear logic. The term language was constructed to be user-friendly. We then described a concrete method of denoting and evaluating quantum programs using the introduced language. Semantically the quantum part of the lambda calculus was shown to resemble the categories known from quantum mechanics. The additional structure the term language yields was successfully incorporated into the categorical semantics.

This thesis made clear how category theory as a mean for connecting various mathematical fields is an irreplaceble tool. It was capable of translating between mathemtical physics, logic and the theory of computation. Linear logic was deemed fruitful enough to describe both the quantum part of the lambda calculus and its classical control. By tracing derivations back to the underlying logic, it was made clear which properties of the calculus were inherited by the linear logic and which were introduced by the specific choice of a term language. Large parts of this thesis are thus also applicable to alternative calculi based on the same logic. The quantum lambda calculus was constructed successfully. It is user-friendly and ready to be employed.

List of Tables

2.1 Multiplicative and additive fragment of linear logic. 23

3.1 Inference rules of the subtyping relation. Here $m, n \in \mathbb{N}$ and we always assume $(m = 0) \vee (n \geq 1)$. 38
3.2 Typing rules. Here A_c denotes the type of constant c (cmp. (3.2)). 39
3.3 Reduction rules for classical control of quantum closures. Here M, N are arbitrary terms and V, W are values. 43
3.4 Reduction rules for quantum data of quantum closures. Here U denotes a unitary gate of arity n. All closures are assumed to be in quantum state $|Q\rangle$. In the first rule the indices $j_i \in \{1, .., n\}$ are all distinct. We write $|Q^i_j\rangle$ for a superposition of states in which $x_i = |j\rangle$, j = 0,1, and α^i, β^i are such that $\alpha^i|Q^i_0\rangle + \beta^i|Q^i_1\rangle = |Q\rangle$. 43
3.5 Congruence rules for quantum closures. Here M, N are arbitrary terms while V is a value. 44
3.6 Axiomatic equivalence: Rules from β-reduction and η-expansion. Here $\ominus$ denotes the symmetric difference of sets, i.e. $S \ominus S' := (S \cup S')\backslash(S \cap S')$. 48
3.7 Axiomatic equivalence: Substitution rules. If in a term M the subterm $let\ x = P\ in\ N$ appears, we imply that x does not appear in the rest of the term M, e.g. in the case (let_1) if $\psi = \langle x, y\rangle$, then x, y do not appear in P. In the following φ and ψ may both stand for x, $\langle x, y\rangle$ and $\star$ (not respectively). 49

Bibliography

[1] S. Abramsky "Computational interpretations of linear logic", *Theoretical Computer Science*, volume 111, issues 1–2, pp. 3-57, 1993.

[2] S. Abramsky, B. Coecke "A categorical semantics of quantum protocols", *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pp. 415-425, Turku, 2004.

[3] S. Abramsky, R. Duncan "A categorical quantum logic", *Mathematical Structures in Computer Science*, vol. 16, issue 3, pp. 469-489, Cambridge University Press: July 2006.

[4] S. Awodey "Category Theory", 2nd edition, Oxford Logic Guides 52, June 2010.

[5] J.C. Baez "Quantum Quandaries: A Category-Theoretic Perspective", *Structural Foundations of Quantum Gravity*, pp. 240-265, retrieved from: `https://math.ucr.edu/home/baez/quantum/quantum.pdf`, last visited: 13.01.23

[6] J.C. Baez, M. Stay "Physics, Topology, Logic and Computation: A Rosetta Stone", in *New Structures for Physics*, pp. 95-174, ed. Bob Coecke, Lecture Notes in Physics vol. 813, Springer, Berlin, 2011.

[7] P.N. Benton "A mixed linear and non-linear logic: Proofs, terms and models". In: *Computer Science Logic* eds. L. Pacholski, J. Tiuryn, Lecture Notes in Computer Science, vol 933, pp. 121–135, Springer, Berlin, Heidelberg: 1995.

[8] G.M. Bierman "What is a categorical model of Intuitionistic Linear Logic?" In: *Typed Lambda Calculi and Applications*, eds. M. Dezani-Ciancaglini, G. Plotkin, Lecture Notes in Computer Science, vol 902, Springer, Berlin, Heidelberg: 1995.

[9] J.R.B. Cockett, R.A.G. Seely "Proof theory for full intuitionistic linear logic, bilinear logic, and mix categories",*Theory and Applications of Categories*, Vol. 3, No. 5, pp. 85–131, 1997.

[10] R. Duncan "Generalized Proof-Nets for Compact Categories with Biproducts" In: *Semantic Techniques in Quantum Computation*, eds. S.Gay, I. Mackie, pp. 70-134, Cambridge, Cambridge University Press: 2009.

[11] R. Duncan "Types for Quantum Computing", Diss. Merton College, Oxford: 2006. Retrieved from: `http://personal.strath.ac.uk/ross.duncan/papers`, last visited: 23.10.23

[12] P. Fu, K. Kishida, P. Selinger "Linear Dependent Type Theory for Quantum Programming Languages: Extended Abstract", in *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 440–453, Association for Computing Machinery, New York: 2020.

[13] J.-Y. Girard "Linear logic", *Theoretical Computer Science*, vol. 50, issue 1, pp. 1-101, 1987.

[14] L.K. Grover "A fast quantum mechanical algorithm for database search" in *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, pp. 212–219, Association for Computing Machinery, New York, 1996.

[15] B.C. Hall "Quantum Theory for Mathematicians", *Graduate Texts in Mathematics*, vol. 267, pp.53-169, 540, Springer: 2013.

[16] O. Malherbe, P. Scott, P. Selinger "Presheaf Models of Quantum Computation: An Outline", In: *Computation, Logic, Games, and Quantum Foundations*, eds. B. Coecke, L. Ong, P. Panangaden, The Many Facets of Samson Abramsky, Lecture Notes in Computer Science, vol 7860. Springer, Berlin, Heidelberg: 2013.

[17] E. Moggi, "Notions of computation and monads", *Information and Computation* volume 93, issue 1, pp. 55-92, 1991.

[18] V. Moretti "Spectral Theory and Quantum Mechanics - Mathematical Foundations of Quantum Theories, Symmetries and Introduction to the Algebraic Formulation", 2nd edition, pp. 107-117, 197-214, Springer: 2017.

[19] S. Negri "A normalizing system of natural deduction for intuitionistic linear logic", *Archive for Mathematical Logic 41*, pp. 789-810, Springer: 2002.

[20] M. Riley "A Bunched Homotopy Type Theory for Synthetic Stable Homotopy Theory", Diss. Wesleyan University: 2022. Retrieved from: `https://mvr.hosting.nyu.edu/pubs/thesis.pdf`, last visited: 27.10.23

[21] P. Selinger "Control categories and duality: On the categorical semantics of the lambda-mu calculus", *Mathematical Structures in Computer Science*, volume 11, issue 2, pp. 207-260, 2001.

[22] P. Selinger, B. Valiron "Quantum Lambda Calculus" in S. Gay, I. Mackie (eds) *Semantic Techniques in Quantum Computation*, pp. 135-172, Cambridge: 2009.

[23] P. Selinger, B.Valiron, "A lambda calculus for quantum computation with classical control" in P. Urzyczyn (eds) *Typed Lambda Calculi and Applications*, pp. 354-368, Lecture Notes in Computer Science, vol 3461, Springer: 2005.

[24] P.W. Shor "Algorithms for quantum computation: discrete logarithms and factoring", *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA, 1994, pp. 124-134.

[25] B. Valiron "Semantics for a Higher Order Functional Programming Language for Quantum Computation", Diss. University of Ottawa: May 2010. Retrieved from: `https://theses.hal.science/tel-00483944#`, last visited: 04.06.2023

[26] A. van Tonder "A Lambda Calculus for Quantum Computation", SIAM Journal on Computing, vol. 33, pp. 1109-1135, 2004.

[27] J. Wickerson "A Very Rough Introduction to Linear Logic", retrieved from: `https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=9308FF5AEB2702B3C0AFD7C702E7D6FF?doi=10.1.1.641.7553&rep=rep1&type=pdf`, last visited: 27.10.2023