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

Simulación del lambda cálculo de matrices de densidad en el lambda cálculo cuántico de Selinger y Valiron

Simulation of the density matrix lamba calculus into the quantum lambda calculus by Selinger and Valiron

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Agustín Borgna

Director: Alejandro Díaz-Caro

Buenos Aires, 2019

El espacio de los estados cuánticos en el campo de la computación cuántica puede ser representado por vectores en un espacio de Hilbert o por matrices de densidad. Selinger y Valiron definieron λ_q en 2005, una extensión cuántica del cálculo lambda que utiliza vectores para representar el estado cuántico y sigue el paradigma de datos cuánticos / control clásico.

El cálculo λ_ρ introducido por Díaz-Caro en 2017, en cambio, describe los estados cuánticos utilizando matrices de densidad. Estas matrices proporcionan una forma de representar estados cuánticos mixtos. Una modificación de este cálculo llamada λ_ρ^o extiende λ_ρ mediante la adición de sumas algebraicas de términos para representar una generalización de las matrices de densidad.

En este trabajo analizamos la relación entre los cálculos definiendo una traducción de λ_ρ a λ_q y su inversa. Usando la traducción probamos la normalización fuerte de λ_ρ. Luego demostramos que las matrices de densidad generalizadas en el cálculo λ_ρ^o son equivalentes a una elección no-determinista entre términos en λ_ρ y definimos una simulación completa de λ_ρ^o en λ_q.

Palabras claves: Cálculo lambda, computación cuántica, matrices de densidad, control clásico.

The space of quantum states in the field of quantum computing can be represented with vectors in a Hilbert space, or with density matrices. Selinger and Valiron defined λ_q in 2005, a quantum extension to the lambda calculus using vectors to represent the quantum states and following the quantum data / classical control paradigm.

The λ_ρ calculus introduced by Diaz-Caro in 2017, on the other hand, describes quantum states using density matrices. These matrices provide a way to represent mixed quantum states. A modification of this calculus called λ_ρ^o extends λ_ρ by adding algebraic sums of terms to represent a generalization of density matrices.

In this thesis we analyze the relationship between the calculi by defining a translation from λ_ρ to λ_q and its left-inverse. Using the translation we prove the strong normalization of λ_ρ. We then show that the generalized density matrices in the λ_ρ^o calculus are equivalent to non-deterministic choices between terms in λ_ρ and define a complete simulation of λ_ρ^o into λ_q.

Keywords: Lambda calculus, quantum computing, density matrices, classical control.

Agradecimientos

A toda mi familia, y especialmente a mis padres/viejos, que estuvieron conmigo durante todo el camino.

A mis amigos, por bancarme aún cuando desaparezca entre exámenes y TPs.

A Pablo, por mostrarme semana a semana el mundo de los cálculos, los tipos, y las demostraciones sobre demostraciones.

A Jano, siempre disponible para resolver todas mis dudas y explicar ideas.

A Beniamino y Charly, por ser jurados de esta tesis y aportar múltiples comentarios.

A mi tío Miguel.

Contents

Introduction **xi**

1 Preliminaries **1**
1.1 Notation . . . 1
1.2 Quantum computing . . . 1
1.2.1 Quantum state . . . 1
1.2.2 Density matrices . . . 4
1.2.3 Models of computation . . . 6
1.3 The simply typed lambda calculus . . . 7

2 Quantum lambda calculi **11**
2.1 λ_q, the linear quantum lambda calculus . . . 11
2.1.1 Quantum closures . . . 12
2.1.2 Rewrite system . . . 13
2.1.3 Reorder and deletion equivalence . . . 15
2.1.4 The *caseof* statement . . . 15
2.2 λ_ρ, a density matrix calculus . . . 16
2.2.1 Rewrite system and reduction strategy . . . 18
2.2.2 Denotational semantics . . . 19
2.3 λ_ρ^o, the generalized λ_ρ . . . 20
2.3.1 Rewrite system . . . 22
2.3.2 Interpretation . . . 22

3 Simulations **25**
3.1 Translation from λ_ρ to λ_q . . . 25
3.1.1 Mixed state purification . . . 25
3.1.2 Translation definition . . . 26
3.1.3 Correctness . . . 29
3.2 Retraction from λ_q to λ_ρ . . . 37
3.3 Translation from λ_ρ^o to λ_q . . . 39
3.3.1 Correctness . . . 40
3.4 Pseudoinverse from λ_ρ to λ_ρ^o . . . 42

4 Conclusions **43**
4.1 Future work . . . 43

Introduction

In the last decade there has been an abundance of research around quantum extensions to the lambda calculus, e.g. [vT04, SV05, PSV14, Zor16, AD17, ADCV17, DCD17]. In all these works, the chosen language used to represent the quantum state has been vectors in a Hilbert space. However, there exists an alternate formulation for quantum mechanics using density matrices. The density matrices provide a way to describe a mixed-state quantum system; that is, a probabilistic set of several possible states. All of the quantum mechanics postulates can be described through such formalism, and therefore all of quantum computing can also be described through it.

[Sel04] introduced a language for quantum flow charts, and an interpretation of this language into a density matrix CPO[1]. After his work, the density matrix language has been widely used in quantum programming languages, e.g. [DP06, FDY11, Yin11, FYY13, YYW17]. Moreover, the "Foundations of Quantum Programming" book [Yin16] is entirely written in the language of density matrices. However, to our knowledge, the only lambda calculus using density matrices is the one introduced in [DC17].

In addition to the distinction of languages based on how they handle quantum states (vectors in a Hilbert space v.s. density matrices), we can also differentiate them by how they consider the control, which can be quantum or classical.

The concept of quantum data / classical control states that the quantum computer runs in a specialized device attached to a classic computer, and that the classical computer instructs the quantum computer on which operations to perform and reads the result after measurements. Many studies have been developed following this paradigm, e.g. [AG05, SV05, GLR$^+$13, PSV14, Zor16]. The idea of having a quantum language where control is classical and data is quantum was described in Knill's QRAM model [Kni96] in 1996. This inspired Selinger's work on quantum programming languages [Sel04], which later induced the creation of a quantum lambda calculus in this paradigm [SV05]. This calculus was the basis for building Quipper [GLR$^+$13], a scalable programming language embedded in Haskell.

Dual to the paradigm of quantum data / classical control, there is the paradigm of quantum control and data. The idea is to provide a computational definition of the notion of vector space and bilinear functions. Quantum control is also commonly used in the area of quantum walks, e.g. [ABN$^+$01, AAKV01]. There are also several high level languages with quantum control, e.g. [AG05, YYF12, YYF14, BP15]. A lambda calculus with quantum control was recently introduced in [DCGMV19] following a long line of work in that direction [ADC12, DCP12, ADCP$^+$14, AD17, ADCV17].

In [DC17] a quantum extension to the lambda calculus, called λ_ρ, is proposed in the quantum data / classical control paradigm, where quantum data are represented by

[1] Complete partial order.

density matrices. Then, in the same work, a modification of this calculus is introduced, called λ_ρ^o, in which the density matrices are generalized to the classical control: that is, after a measurement, all possible results are taken in a kind of generalized density matrix of programs. The control is not quantum, since it is not possible to superpose programs. However, when considering the density matrix of the mixed state of programs produced by a measurement, the control is not classic either. This new paradigm can be identified as weak quantum control. Thus, λ_ρ and λ_ρ^o become the first quantum calculi to weakly relate the two paradigms.

In this thesis we propose a translation between λ_ρ and the highly developed quantum calculus λ_q, introduced by Selinger and Valiron in [SV05], and define a left-inverse. We also take advantage of it to prove strong normalization in λ_ρ. We then define a a translation from λ_ρ^o to λ_ρ, showing that the weak quantum control can be modelled by the classical control. The composition of both translations results in a simulation of the λ_ρ^o calculus in λ_q.

The purpose of this work is then to show that the calculi λ_ρ and λ_ρ^o are equivalent to the Selinger-Valiron calculus.

The remainder of this work has the following structure:

- In Chapter 1, we go over the preliminary concepts used through this document. We give a swift introduction to quantum mechanics and define the simply typed lambda calculus.
- In Chapter 2, we introduce the quantum extensions to the lambda calculus λ_ρ, λ_ρ^o, and λ_q.
- In Chapter 3, we define the translations between the calculi, and prove their soundness.
- In Chapter 4, we end with a discussion of our results and a proof of strong normalization for λ_ρ.

Chapter 1

Preliminaries

1.1 Notation

As usual, $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{R}$ and $\mathbb{C}$ denote the natural numbers, natural numbers including zero, real numbers, and complex numbers, respectively.

We use the *Dirac notation* to represent unitary vectors in the Hilbert space $\mathbb{C}^2$. The vectors $\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$ composing the canonical basis are denoted with the *kets* $|0\rangle$ and $|1\rangle$ respectively. Another commonly referenced vectors are $|+\rangle = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$ and $|-\rangle = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix}1\\-1\end{smallmatrix}\right)$. A *bra* represents the Hermitian conjugate of a ket, also denoted with a $\dagger$, $\langle\phi| = |\phi\rangle^\dagger$. Notice that $\langle\phi \mid \psi\rangle$ corresponds to the inner product between the vectors $|\phi\rangle$ and $|\psi\rangle$ and $|\phi\rangle\langle\psi|$ corresponds to the outer product. Two vectors $|\phi\rangle$ and $|\psi\rangle$ in the Hilbert spaces V and W may be combined into a vector $|\phi\psi\rangle$ in a third space $V \otimes W$ by the tensor product $|\phi\psi\rangle = |\phi\rangle \otimes |\psi\rangle$.

1.2 Quantum computing

We shall not give a thorough description of quantum computing in this document. There have been a number of great books published about it, e.g. [NC10, KLM07, Mer07]. However, we will briefly review the basis of the theory.

A classical computer operates by applying discrete transformations to an evolving bit state. At any point in time, this state can be described deterministically with the binary valuation for each bit. We can *measure* this state without modifying it, and overwrite it with any values we desire.

A quantum computer is a whole different story, as we shall see in this section.

1.2.1 Quantum state

The state of a quantum computer is composed by units called *quantum bits*, or *qubits* for short. As with classical bits, which can be in either the state 0 or 1, qubits may be in two states we call $|0\rangle$ and $|1\rangle$. But the qubits can also be in any complex linear combination or *superposition* $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Thus, a qubit is a vector in the Hilbert space $\mathbb{C}^2$.

An ensemble of n qubits can be represented by a normalized vector in the Hilbert space $\mathbb{C}^{2^n} = \bigotimes_{i=1}^{n} \mathbb{C}^2$. The canonical basis of this space is described by the combination of the basis vectors for each qubit. For $n = 2$, this corresponds to $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. We use $||\psi\rangle|$ to denote the number of qubits in a state $|\psi\rangle$.

Some quantum states in a composed state cannot be written as the tensor product between states of each individual qubit. For example, the *Bell state*

$$\beta_{00} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

The qubits in this state are said to be *entangled*.

Evolution

The evolution of a *closed* quantum system, a system that does not interact with an external physical system, can be described as a succession of discrete steps as *unitary operators*, matrices $U \in \mathbb{C}^n$ such that $U^\dagger U = I$. An operator in C^{2^n} is also called an n-ary *quantum gate*, since it operates over a state of n qubits. By definition, this operation is always invertible.

Example 1.2.1 *The Hadamard gate is a single-qubit unitary operator that maps the states $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$ respectively. It is defined as follows:*

$$\mathsf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Example 1.2.2 *Another commonly referenced gate is the binary* CNOT*, or controlled NOT gate. It performs a* NOT *operation on the second qubit only when the first qubit is in the state $|1\rangle$, and leaves it unchanged otherwise.*

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Remark 1.2.3 *An important consequence of this definition is the no-cloning theorem.*

Suppose that we had a state $|s\rangle$ and some unitary operator U that was able to copy two particular states ϕ and ψ:

$$\begin{aligned} U|\phi s\rangle &= |\phi\phi\rangle \\ U|\psi s\rangle &= |\psi\psi\rangle \end{aligned}$$

Then $\langle U\phi s \mid U\psi s\rangle = \langle \phi\phi \mid \psi\psi\rangle = (\langle \phi \mid \psi\rangle)^2$. But, on the other hand, $\langle U\phi s \mid U\psi s\rangle = \langle \phi s \mid \psi s\rangle = \langle \phi \mid \psi\rangle$. Then, we have $(\langle \phi \mid \psi\rangle)^2 = \langle \phi \mid \psi\rangle$. That is, ϕ and ψ are either equal or orthogonal. Therefore a cloning unitary operator can only clone orthogonal quantum states, there is no universal cloning machine.

Formally, there is no quantum gate U and quantum state $|\phi\rangle \in \mathbb{C}^n$ such that for any $|\psi\rangle \in \mathbb{C}^n$, $U|\psi\phi\rangle = |\psi\psi\rangle$.

Measurement

The other way a quantum state may evolve is through interaction with an external physical system. This process is called a *measurement*. The external observer is able to obtain some information from the actual state of the quantum system, while inevitably disturbing it.

The measurement is described as a collection of *measurement operators* $\{M_i\}_{i=1}^m$ satisfying the property $\sum_{i=1}^m M_i^\dagger M_i = I$.

When the measurement is performed over a state $|\psi\rangle$, a single operator is chosen randomly with probability:

$$p_i = \langle\psi| M_i^\dagger M_i |\psi\rangle$$

The index k of the chosen operator M_k is called the *result* of the measurement, and is known by the external observer. This process *collapses* the system into a new state $|\psi'\rangle$:

$$|\psi'\rangle = \frac{M_k |\psi\rangle}{\sqrt{\langle\psi| M_k^\dagger M_k |\psi\rangle}}$$

Notice that this operation is idempotent. If a measurement is performed over a quantum state, further measurements using the same set of measurement operators will always yield the same result.

A single collection of measurement operators suffices to perform any desired measurement when combined with a unitary operator. Through this document we use the set of operators derived from the canonical basis,

$$\{|0\dots00\rangle\langle0\dots00|, |0\dots01\rangle\langle0\dots01|, \dots, |1\dots11\rangle\langle1\dots11|\}.$$

We write a measurement operation of a state $|\psi\rangle$ over the canonical basis as $\pi|\psi\rangle$.

Example 1.2.4 *Consider a measurement operation of the state* $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ *over the canonical basis,* $\pi|+\rangle$*. The associated measurement operators are:*

$$M_0 = |0\rangle\langle0| \qquad M_1 = |1\rangle\langle1|$$

The probabilities for the measurement operators are:

$$p_0 = \langle+| M_0^\dagger M_0 |+\rangle = \frac{1}{2} \qquad p_1 = \langle+| M_1^\dagger M_1 |+\rangle = \frac{1}{2}$$

And the possible final states are:

$$|\psi_0'\rangle = \frac{M_0 |+\rangle}{\sqrt{\langle+| M_0^\dagger M_0 |+\rangle}} = |0\rangle \qquad |\psi_1'\rangle = \frac{M_1 |+\rangle}{\sqrt{\langle+| M_1^\dagger M_1 |+\rangle}} = |1\rangle$$

Quantum postulates

The given definitions of a quantum system are independent from the specific physical system used by the quantum computer. The properties of the qubit and its behaviour are based on a set of quantum postulates that let us abstract over the hardware implementation of the quantum computer when constructing the theory around it.

The previous notions follow the four postulates of quantum mechanics, as defined in [NC10]:

Postulate 1: Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

The first postulate does not define the specific Hilbert space to be used. For the states presented in this section we chose the commonly used $\mathbb{C}^{2^n}$ spaces. In Subsection 1.2.2 we define an alternate realization of the postulates using density matrices.

Postulate 2: The evolution of a *closed* quantum system is described by a *unitary transformation*. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2, $|\psi'\rangle = U|\psi\rangle$.

Postulate 3: Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, then the probability that result m occurs is given by

$$p_m = \langle\psi| M_m^\dagger M_m |\psi\rangle$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}}.$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I.$$

The completeness equation expresses the fact that probabilities sum to one:

$$1 = \sum_m p_m = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle .$$

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.

1.2.2 Density matrices

In the previous Subsection we formulated quantum mechanics based on vectors in a complex Hilbert space, but the first postulate of quantum mechanics also allows for an alternative definition using *density operators* (or *density matrices*).

Suppose we have a quantum system that might be in a number of states $|\psi_i\rangle$ with probability p_i. An *ensemble of quantum bits* is a set $\{p_i, |\psi_i\rangle\}_i$, with $\sum_i p_i = 1$. The density operator for this state is defined as

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

This matrix is Hermitian, positive-semidefinite, and has trace one.

If a density matrix can be decomposed into a single state $|\psi\rangle$ such that $\rho = |\psi\rangle\langle\psi|$ we call it a *pure-state* density matrix. This is equivalent to asking for the square of the matrix to have trace one, $\mathsf{tr}(\rho^2) = 1$. If a density matrix is not in a pure state we say it corresponds to a *mixed state*.

Example 1.2.5 *The pure density matrix corresponding to the state $|+\rangle$ can be written as:*

$$\rho = |+\rangle\langle+| = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

Example 1.2.6 *We can write the mixed state density matrix with no information about the state, starting from the ensemble $\{(\frac{1}{2}, |0\rangle), (\frac{1}{2}, |1\rangle)\}$. That is, a state where each posible state in a basis has the same probability,*

$$\rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}.$$

Notice that any other ensemble of basis vectors with equal probabilities yields the same density matrix,

$$\rho' = \frac{1}{2}|+\rangle\langle+| + \frac{1}{2}|-\rangle\langle-| = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}.$$

The equivalence corresponds to the fact that both states are physically indistinguishable from each other.

Quantum postulates reformulation

The postulates of quantum mechanics can then be reformulated to use density matrices, as defined in [NC10]:

> **Postulate 1:** Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the *state space* of the system. The system is completely described by its *density operator*, which is a positive operator ρ with trace one, acting on the state space of the system. If a quantum system is in the state ρ_i with probability p_i, then the density operator for the system is $\sum_i p_i\rho_i$.
>
> **Postulate 2:** The evolution of a *closed* quantum system is described by a *unitary transformation*. That is, the state ρ of the system at time t_1 is related to the state ρ' of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2, $\rho' = U\rho U^\dagger$.
>
> **Postulate 3:** Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is ρ immediately before the measurement, then the probability that result m occurs is given by
>
> $$p_m = \mathsf{tr}(M_m^\dagger M_m \rho)$$

and the state of the system after the measurement is

$$\frac{M_m \rho M_m^\dagger}{\mathsf{tr}(M_m^\dagger M_m \rho)}.$$

The measurement operators satisfy the *completeness equation*,

$$\sum_m M_m^\dagger M_m = I.$$

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $\rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n$.

The reduced density operator

The density operator can be used to describe *subsystems* of a composite quantum system, via what is called the *reduced density operator*. If two quantum systems A and B are composed into a single system $A \otimes B$ and its state can be described by the density matrix ρ^{AB}, then the reduced density operator for system A is defined as

$$\rho^A = \mathsf{tr}_B(\rho^{AB}),$$

where tr_B is the *partial trace* over the system B. We later use this operation when defining a method for finding pure states containing a given mixed state, in Section 3.1.

Example 1.2.7 *Let $A = B = \mathbb{C}^2$ and let $\rho \in A \otimes B$ be the pure state density matrix corresponding to the Bell state β_{00},*

$$\rho = \beta_{00}^\dagger \beta_{00} = \frac{|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|}{2} = \begin{bmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 \end{bmatrix}$$

Now, consider the reduced density operator for system A,

$$\rho' = \mathsf{tr}_B(\rho) = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} = \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}$$

It corresponds to the no-information mixed state density matrix described in Example 1.2.6.

1.2.3 Models of computation

There have been two main paradigms related to how a quantum computer would handle the control and data flow in a program.

The *classical control-quantum data* paradigm, attributed to Knill [Kni96], proposes that a quantum computer would be based on a classical machine with an attached quantum device. The flow of the program would be controlled by the classical part, and the quantum coprocessor would maintain an internal quantum state and allow for quantum operations to be run on it, returning the result of any measurement.

The *quantum control-quantum data* counterpart proposes a machine where the flow of the program is directly controlled by the quantum state, allowing for different branches of the program to be executed in superposition. This control paradigm is commonly used in the field of quantum walks.

The lambda calculi introduced in Sections 2.1 and 2.2 are based on the classical control paradigm. In Section 2.3 we introduce a third calculus that considers outcomes of a measurement in a kind of generalized density matrix of arbitrary terms. We call this type of control *probabilistic control*, or *weak quantum control*, since it does not allow superpositions of terms.

1.3 The simply typed lambda calculus

The lambda calculus was introduced by Alonzo Church in the 1930s [Chu36] as a formalization of computable functions. Church later defined a typed interpretation in [Chu40] called *simply typed λ-calculus*. In this section we introduce the general notions of the simply typed λ-calculus, and some other definitions to be used through this document.

The set of terms Λ in the simply typed λ-calculus are defined recursively as follows:

$$t ::= x \mid t\,t \mid \lambda x.t$$

where x is a variable from an infinite denumerable set $\mathcal{V}$.

An occurrence of a variable x is said to be *bound* in t if appears in a subterm $\lambda x.t'$. If the variable is not bound, it is called a *free variable* of t. We write the set of free variables of a term $\mathsf{FV}(t)$.

A bound variable in t may be renamed to an unused variable by *α-conversion* while maintaining the meaning of the term. We consider the terms under α-equivalence and adopt the *Barendregt Convention* [Bar84], which stipulates that the set of free variables is disjoint from the set of bound variables, and that each subterm $\lambda x.t'$ binds a different variable.

The simply typed λ-calculus defines a set of *reduction* or *rewrite rules* that determines the relation $t \to r$, given in Table 1.1. $t[r/x]$ represents a *substitution*, replacing all the occurrences of the variable x in t by r.

$$(\lambda x.\ t)r \to t[r/x] \qquad \frac{t \to t'}{\lambda x.t \to \lambda x.t'} \qquad \frac{r \to r'}{tr \to tr'} \qquad \frac{t \to t'}{tr \to t'r}$$

Where $t[r/x]$ represents the substitution of each free variable x by the term r.

Table 1.1: Rewrite system for the simply typed λ-calculus

We also define a set of *value terms* based on the reduction rules, which corresponds to the terms that cannot be further reduced:

$$v ::= x \mid \lambda x.v$$

For certain terms the rewrite rules allow for more than one possible reduction, as seen in Example 1.3.1.

Example 1.3.1 *The term* $t = (\lambda x.x\,((\lambda y.y)\,x_2))\,x_1$ *reduces to two different terms:*

$$t \to x_1\,((\lambda y.y)\,x_2) \text{ and } t \to (\lambda x.x\,x_2)\,x_1.$$

It is possible to modify the reduction rules to follow a certain *reduction strategy*, which deterministically defines at most one possible reduction for each term.

A commonly used reduction strategy is called *call-by-value* [Plo75]. This strategy mimics the behaviour of most imperative programming languages, in that it forces each argument of a function to be fully evaluated before running it. A variation of this strategy, called *weak* call-by-value, also disallows internal reductions inside a lambda abstraction.

In Table 1.2 we redefine the reduction rules using the weak call-by-value strategy. The set of value terms is redefined as:

$$v ::= x \mid \lambda x.t$$

$$(\lambda x.\ t)v \to t[v/x] \qquad \frac{r \to r'}{tr \to tr'} \qquad \frac{t \to t'}{tv \to t'v}$$

Where v is a value.

Table 1.2: Rewrite system for the simply typed λ-calculus with a call-by-value strategy

The simply typed lambda calculus defines a typed interpretation of its terms. The set of terms are defined recursively as:

$$A ::= \alpha \mid A \to A$$

where α belongs to a fixed set of *base types*.

A *typing context* Γ is an unordered set of *typing assumptions* $x : A$, attributing the type A to the term x. A *typing judgement* $\Gamma \vdash t : A$ states that the term t is of type A in the context Γ. This judgement is derived via a set of *typing rules*. The set of typing rules for the simply typed λ-calculus is defined in Table 1.3. We define the domain of a typing context as the set of variables in its typing assumptions, $\mathsf{dom}(\{x_i : A_i\}_i) = \{x_i\}_i$.

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

Table 1.3: Typing rules for the simply typed λ-calculus

A typing system may admit a number of structural rules, either explicitly defined or as lemmas.

The *weakening* rule (Definition 1.3.2) allows us to ignore some typing assumptions from the context, discarding the ones we don't need.

The *contraction* rule (Definition 1.3.3) allows us to use a typing assumptions multiple times for separate branches of the typing judgement. In practice, this lets us reference a bound variable multiple times in a well-typed term.

Definition 1.3.2 (Weakening)

$$\frac{\Gamma \vdash t : A}{\Gamma, x : B \vdash t : A}\ \texttt{weakening}$$

Definition 1.3.3 (Contraction)

$$\frac{\Gamma, x : B, y : B \vdash t : A}{\Gamma, x : B \vdash t[x/y] : A}\ \texttt{contraction}$$

Notice that the two rules can be proved as derivable in the simply typed lambda calculus.

A typing system that admits weakening but not contraction is called *affine*. We use this kind of typing system for the calculi presented in Sections 2.1, 2.2, and 2.3, since it allows us to have variables linked to qubits, while honoring the no-cloning property.

A second kind of type system, called *linear*, does not admit contraction nor weakening. That is, each variable in the context must be used exactly once.

By convention, function types in affine and linear systems are denoted with a *lollipop* arrow $\multimap$, instead of the traditional arrow.

Chapter 2

Quantum lambda calculi

2.1 λ_q, the linear quantum lambda calculus

The Selinger-Valiron quantum lambda calculus (λ_q) has been introduced in [SV05]. This calculus was the first quantum extension to the lambda calculus to use the classical control/quantum data paradigm. λ_q was the basis on which the quantum programming language Quipper [GLR+13] was created.

The calculus follows the idea of having a quantum state separated from the term, where variables are used as pointers to specific qubits in the state.

In [SV08], Selinger and Valiron defined a fragment of this calculus where all the variables are linearly typed, in contrast to the original calculus that used an affine system with an *unrestricted* type constructor !. They also added a non-terminating term Ω. Through this document we use a combination of both definitions: We use the fragment without the unrestricted type, but maintaining the weakening rule from the original affine system. We also omit the non-terminating term. This calculus uses a call-by-name strategy.

The calculus λ_q extends the simply typed lambda calculus with booleans and products, and adds specific terms to deal with a quantum register. We refer to the set of terms as Λ_q, and define them as follows:

$$\begin{gathered} t ::= x \mid tt \mid \lambda x.\ t \mid \mathsf{if}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t \mid 0 \mid 1 \mid \mathsf{meas} \mid \mathsf{new} \mid \\ U \mid * \mid \langle t, t\rangle \mid \mathsf{let}\ \langle x, y\rangle = t\ \mathsf{in}\ t \mid \mathsf{let}\ * = t\ \mathsf{in}\ t \end{gathered}$$

The new term represents the process of initializing a new qubit in the quantum register according to a boolean variable. meas, in turn, measures a single qubit from the quantum register over the canonical base and returns the boolean result. And finally, U may be any unitary operator that can get applied to a tuple of qubits.

The shorthand notation $\langle t_1, t_2, \dots t_n\rangle$ is equivalent to $\langle t_1, \langle t_2, \dots\rangle\rangle$.

Types in λ_q are defined as follows:

$$A ::= \mathsf{bit} \mid \mathsf{qbit} \mid A \multimap A \mid \top \mid A \otimes A$$

We refer to the set of types as Π_q. We use the notation $\mathsf{qbit} \otimes \cdots \otimes \mathsf{qbit} = \mathsf{qbit}^n$ and $\mathsf{bit} \otimes \cdots \otimes \mathsf{bit} = \mathsf{bit}^n$.

Example 2.1.1 *The following term emulates the process of flipping a fair coin and choosing between two terms t and r based on the result. The unitary operator H is the Hadamard gate as defined in Section 1.2.*

$$\mathsf{if}\ \mathsf{meas}(\mathsf{H}(\mathsf{new}\ 0))\ \mathsf{then}\ t\ \mathsf{else}\ r$$

The typing rules for λ_q are defined in Table 2.1. We write c for an arbitrary constant of the language, i.e. 0, 1, meas, new, U, or $*$. We call A_c the type of the constant c, defined as follows:

$$A_0 = A_1 = \mathsf{bit} \qquad A_{\mathsf{meas}} = \mathsf{qbit} \multimap \mathsf{bit} \qquad A_{\mathsf{new}} = \mathsf{bit} \multimap \mathsf{qbit}$$
$$A_{U^n} = \mathsf{qbit}^n \multimap \mathsf{qbit}^n \qquad * = \top$$

Since the type system is affine, it admits the weakening rule (Lemma 2.1.2).

$$\frac{}{\Delta, x : A \vdash x : A}\,\mathsf{ax}_1 \qquad \frac{}{\Delta, \vdash c : A_c}\,\mathsf{ax}_2 \qquad \frac{\Gamma_1 \vdash t : \mathsf{bit} \quad \Gamma_2 \vdash r : A \quad \Gamma_2 \vdash s : A}{\Gamma_1, \Gamma_2 \vdash \mathsf{if}\ t\ \mathsf{then}\ r\ \mathsf{else}\ s : A}\,\mathsf{if}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\,\multimap_I \qquad \frac{\Gamma_1 \vdash t : A \multimap B \quad \Gamma_2 \vdash r : A}{\Gamma_1, \Gamma_2 \vdash tr : B}\,\multimap_E$$

$$\frac{\Gamma_1 \vdash t_1 : A_1 \quad \Gamma_2 \vdash t_2 : A_2}{\Gamma_1, \Gamma_2 \vdash \langle t_1, t_2 \rangle : A_1 \otimes A_2}\,\otimes_I \qquad \frac{\Gamma_1 \vdash r : A_1 \otimes A_2 \quad \Gamma_2, x_1 : A_1, x_2 : A_2 \vdash t : A}{\Gamma_1, \Gamma_2 \vdash \mathsf{let}\ \langle x_1, x_2 \rangle = r\ \mathsf{in}\ t : A}\,\otimes_E$$

$$\frac{}{\vdash \star : \top}\,\top_I \qquad \frac{\Gamma_1 \vdash r : \top \quad \Gamma_2 \vdash t : A}{\Gamma_1, \Gamma_2 \vdash \mathsf{let}\ \star = r\ \mathsf{in}\ t : A}\,\top_E$$

Table 2.1: Typing rules for λ_q

Lemma 2.1.2 (Weakening) *If $\Gamma \vdash t : A$ and $x \notin \mathsf{FV}(t)$, then $\Gamma, x : B \vdash t : A$.* □

Example 2.1.3 *Given a typing context Γ such that $\Gamma \vdash t : A$ and $\Gamma \vdash r : A$, we can write the following typing derivation for the coin-flipping term from Example 2.1.1.*

Let Π_{H} be the following derivation

$$\frac{\dfrac{}{\vdash \mathsf{H} : \mathsf{qbit} \multimap \mathsf{qbit}}\,\mathsf{ax}_2 \qquad \dfrac{\dfrac{}{\vdash \mathsf{new} : \mathsf{bit} \multimap \mathsf{qbit}}\,\mathsf{ax}_2 \quad \dfrac{}{\vdash 0 : \mathsf{bit}}\,\mathsf{ax}_2}{\vdash \mathsf{new}\ 0 : \mathsf{qbit}}\,\multimap_E}{\vdash \mathsf{H}(\mathsf{new}\ 0) : \mathsf{qbit}}\,\multimap_E$$

then

$$\frac{\dfrac{\dfrac{}{\vdash \mathsf{meas} : \mathsf{qbit} \multimap \mathsf{bit}}\,\mathsf{ax}_2 \quad \Pi_{\mathsf{H}}}{\vdash \mathsf{meas}(\mathsf{H}(\mathsf{new}\ 0)) : \mathsf{bit}}\,\multimap_E \quad \Gamma \vdash t : A \quad \Gamma \vdash r : A}{\Gamma \vdash \mathsf{if}\ \mathsf{meas}(\mathsf{H}(\mathsf{new}\ 0))\ \mathsf{then}\ t\ \mathsf{else}\ r : A}\,\mathsf{if}$$

2.1.1 Quantum closures

A program in λ_q is composed by a term where all the free variables correspond to *pointers* to qubits in an external quantum state.

A *program state* (also called a *quantum closure*) is represented by a triple $[Q, L, M] \in \mathcal{C}_q$, where

- Q is a normalized vector of $\bigotimes_{i=0}^{n-1} \mathbb{C}^2$ for some $n \geq 0$.

- M is a lambda term.
- L is an injective *linker* function from a set $\mathcal{V}$ of variables to $\{0, \ldots, n-1\}$.

Since linked variables can be freely renamed we assume no two closures have the same linked variables.

Given two quantum states $[Q_1, L_1, M_1]$ and $[Q_2, L_2, M_2]$ we denote $L_1 \cup L_2$ the union of the two (disjoint) linker functions, assuming the target qubit vector is the result of the tensor product $Q_1 \otimes Q_2$.

$$(L_1 \cup L_2)(x) = \begin{cases} L_1(x) & \text{if } x \in \ domain(L_1) \\ L_2(x) + |Q_1| & \text{if } x \in \ domain(L_2) \end{cases}$$

The following functions allow us to refer to the different parts of the *states*.

$$\begin{aligned} \mathrm{st}([Q, L, M]) &= Q \\ \mathrm{lnk}([Q, L, M]) &= L \\ \mathrm{term}([Q, L, M]) &= M \end{aligned}$$

We extend the notion of substitution to closures as follows

$$Q[R/x] = [\mathrm{st}(Q) \otimes \mathrm{st}(R),\ \mathrm{lnk}(Q) \cup \mathrm{lnk}(R),\ \mathrm{term}(Q)[\mathrm{term}(R)/x]]$$

where $Q, R \in \mathcal{C}_q$.

A quantum closure Q is *well-typed* of type A in a typing context Γ (written $\Gamma \vdash Q : A$) if $\mathsf{dom}(\mathrm{lnk}(Q)) \cap \mathsf{dom}(\Gamma) = \emptyset$, $\mathsf{FV}(\mathrm{term}(Q)) \setminus \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(\mathrm{lnk}(Q))$, and $\Gamma, x_1 : \mathsf{qbit}, \ldots, x_n : \mathsf{qbit} \vdash Q : A$ is a valid type judgement, where $\{x_1, \ldots, x_n\} = \mathsf{FV}(\mathrm{term}(Q)) \setminus \mathsf{dom}(\Gamma)$. That is, Q is well-typed if the variables used as qubit pointers are not in Γ and assigning them the type qbit results in $\mathrm{term}(Q)$ being well-typed under Γ.

Finally, we call a quantum closure Q a *program* if $\vdash Q : A$ is a valid type judgement for some type A.

Example 2.1.4 *The coin example in Example 2.1.1 can be written as a quantum closure with an already-initialized state $|+\rangle$ as follows:*

$$[\left|+\right\rangle,\ \{x_0 \mapsto 0\},\ \mathsf{if\ meas}\ x_0\ \mathsf{then}\ t\ \mathsf{else}\ r]$$

Notice that the closure is a program only if t and r are closed terms.

2.1.2 Rewrite system

The calculus λ_q uses a probabilistic reduction system to model the behaviour of the measurement operation. In a probabilistic reduction system, a reduction step may reduce to a number of different terms, based on a given probability distribution. When defining the rewrite rules, the associated probability p is written besides the reduction arrow, $\hookrightarrow_p$.

λ_q uses a weak call-by-value reduction strategy. The rewrite rules are presented in Table 2.2.

The set of *term values* V_q is defined as follows:

$$v ::= x \mid \lambda x.\ t \mid 0 \mid 1 \mid meas \mid new \mid U \mid * \mid \langle v, v \rangle$$

A *closure value* is a quantum closure of the form $[Q, L, v]$ where $v \in V_q$.

The *trace* of a term is the probabilistic tree of all its possible derivations.

$$[Q,\, L,\, (\lambda x.\ M)v] \hookrightarrow_1 [Q,\, L,\, M[v/x]]$$
$$[Q,\, L,\, \mathsf{let}\ \langle x_1, x_2\rangle = \langle v_2, v_1\rangle\ \mathsf{in}\ N] \hookrightarrow_1 [Q,\, L,\, N[v_1/x_1, v_2/x_2]]$$
$$[Q,\, L,\, \mathsf{let}\ * = *\ \mathsf{in}\ N] \hookrightarrow_1 [Q,\, L,\, N]$$
$$[Q,\, L,\, \mathsf{if}\ 0\ \mathsf{then}\ M\ \mathsf{else}\ N] \hookrightarrow_1 [Q,\, L,\, N]$$
$$[Q,\, L,\, \mathsf{if}\ 1\ \mathsf{then}\ M\ \mathsf{else}\ N] \hookrightarrow_1 [Q,\, L,\, M]$$

If w is a fresh variable:

$$[Q,\, |x_1,\dots,x_n\rangle\,,\, \mathsf{new}\ 0] \hookrightarrow_1 [Q \otimes |0\rangle\,,\, |x_1,\dots,x_n,w\rangle\,,\, w]$$
$$[Q,\, |x_1,\dots,x_n\rangle\,,\, \mathsf{new}\ 1] \hookrightarrow_1 [Q \otimes |1\rangle\,,\, |x_1,\dots,x_n,w\rangle\,,\, w]$$

If Q' is the result of applying U to the qubits $L(x_1),\dots,L(x_n)$ in Q:

$$[Q,\, L,\, U\ \langle x_1,\dots,x_n\rangle] \hookrightarrow_1 [Q',\, L,\, \langle x_1,\dots,x_n\rangle]$$

If $Q_j^b \in \mathbb{C}^{2^{i-1}}$, $\tilde{Q}_j^b \in \mathbb{C}^{2^{n-i}}$, $Q = \sum_j Q_j^0 \otimes \alpha_j\,|0\rangle \otimes \tilde{Q}_j^0 + \sum_j Q_j^1 \otimes \beta_j\,|1\rangle \otimes \tilde{Q}_j^1$, $\alpha = \sum_j \alpha_j$, and $\beta = \sum_j \beta_j$:

$$[Q,\, |x_1,\dots,x_n\rangle\,,\, \mathsf{meas}\ x_i] \hookrightarrow_{|\alpha|^2} [\sum_j Q_j^0 \otimes \tilde{Q}_j^0,\, |x_1,\dots,x_{i-1},x_{i+1},\dots,x_n\rangle\,,\, 0]$$
$$[Q,\, |x_1,\dots,x_n\rangle\,,\, \mathsf{meas}\ x_i] \hookrightarrow_{|\beta|^2} [\sum_j Q_j^1 \otimes \tilde{Q}_j^1,\, |x_1,\dots,x_{i-1},x_{i+1},\dots,x_n\rangle\,,\, 1]$$

$$\frac{[Q,\, L,\, N] \hookrightarrow_p [Q',\, L',\, N']}{[Q,\, L,\, MN] \hookrightarrow_p [Q',\, L',\, MN']} \qquad \frac{[Q,\, L,\, M] \hookrightarrow_p [Q',\, L',\, M']}{[Q,\, L,\, Mv] \hookrightarrow_p [Q',\, L',\, M'v]}$$

$$\frac{[Q,\, L,\, M_1] \hookrightarrow_p [Q',\, L',\, M_1']}{[Q,\, L,\, \langle M_1, M_2\rangle] \hookrightarrow_p [Q',\, L',\, \langle M_1', M_2\rangle]} \qquad \frac{[Q,\, L,\, M_2] \hookrightarrow_p [Q',\, L',\, M_2']}{[Q,\, L,\, \langle v_1, M_2\rangle] \hookrightarrow_p [Q',\, L',\, \langle v_1, M_2'\rangle]}$$

$$\frac{[Q,\, L,\, P] \hookrightarrow_p [Q',\, L',\, P']}{[Q,\, L,\, \mathsf{if}\ P\ \mathsf{then}\ M\ \mathsf{else}\ N] \hookrightarrow_p [Q',\, L',\, \mathsf{if}\ P'\ \mathsf{then}\ M\ \mathsf{else}\ N]}$$

$$\frac{[Q,\, L,\, M] \hookrightarrow_p [Q',\, L',\, M']}{[Q,\, L,\, \mathsf{let}\ \langle x_1, x_2\rangle = M\ \mathsf{in}\ N] \hookrightarrow_p [Q',\, L',\, \mathsf{let}\ \langle x_1, x_2\rangle = M'\ \mathsf{in}\ N]}$$

$$\frac{[Q,\, L,\, M] \hookrightarrow_p [Q',\, L',\, M']}{[Q,\, L,\, \mathsf{let}\ * = M\ \mathsf{in}\ N] \hookrightarrow_p [Q',\, L',\, \mathsf{let}\ * = M'\ \mathsf{in}\ N]}$$

Table 2.2: Rewrite system for λ_q

Example 2.1.5 *The following trace corresponds to a closure based on the coin-flipping term from Example 2.1.1.*

$$\begin{array}{c}
[*, \varnothing, \mathsf{if\ meas(H(new\ 0))\ then}\ t\ \mathsf{else}\ r] \\
\downarrow_1 \\
[|0\rangle, \{x_0 \mapsto 0\}, \mathsf{if\ meas\ (H}\ x_0)\ \mathsf{then}\ t\ \mathsf{else}\ r] \\
\downarrow_1 \\
[|+\rangle, \{x_0 \mapsto 0\}, \mathsf{if\ meas}\ x_0\ \mathsf{then}\ t\ \mathsf{else}\ r] \\
\tfrac{1}{2}\swarrow \qquad \searrow\tfrac{1}{2} \\
[*, \varnothing, \mathsf{if}\ 1\ \mathsf{then}\ t\ \mathsf{else}\ r] \quad [*, \varnothing, \mathsf{if}\ 0\ \mathsf{then}\ t\ \mathsf{else}\ r] \\
\downarrow_1 \qquad\qquad\qquad\qquad \downarrow_1 \\
[*, \varnothing, t] \qquad\qquad\qquad\qquad [*, \varnothing, r]
\end{array}$$

2.1.3 Reorder and deletion equivalence

We define a notion of equivalence between quantum closures, allowing for reordering the qubits in the state and dropping the unused ones. The definition is given in two parts:

Given two states $X = [Q_x, L_x, M_x]$ and $Y = [Q_y, L_y, M_y]$,

- $X \approx_{delete} Y$ iff there exists a state Q_A such that $M_x = M_y$, $Q_x = Q_A \otimes Q'_x$, $Q_y = Q_A \otimes Q'_y$, $\mathrm{FV}(M_x) \subseteq \mathsf{dom}(L_x) \cup \mathsf{dom}(L_y)$, and $\forall x \in \mathrm{FV}(M_x), L_x(x) = L_y(x) < |Q_A|$.
- $X \approx_{swap} Y$ iff $(Q_x, L_x) =_\alpha (Q_y, L_y)$, using the α equivalence for quantum contexts described in [SV08] and $M_x =_\alpha M_y$.

Finally we define $\approx$ as the transitive reflexive and symmetric closure of $\approx_{delete} \cup \approx_{swap}$. In this thesis we consider the quantum closures modulo $\approx$-equivalence.

Example 2.1.6 *Let X be a quantum closure with unused qubits in its state,*
$X[|0110\rangle \otimes |\psi\rangle, \{x_0 \mapsto 0, \dots, x_3 \mapsto 3\}, \langle x_0, x_1, x_2, x_3\rangle]$
Then the extra qubits can be discarded:
$Y[|0110\rangle, \{x_0 \mapsto 0, \dots, x_3 \mapsto 3\}, \langle x_0, x_1, x_2, x_3\rangle] \approx_{delete} X$

Example 2.1.7 *Consider a quantum closure with a two-qubit state,*
$X = [|\phi\rangle \otimes |\psi\rangle, \{x_0 \mapsto 0, x_1 \mapsto 1\}, \langle x_0, x_1\rangle]$.
It is swap-equivalent to a similar state with an inverted qubit order:
$Y = [|\psi\rangle \otimes |\phi\rangle, \{x_0 \mapsto 1, x_1 \mapsto 0\}, \langle x_0, x_1\rangle] \approx_{swap} X$.

Remark 2.1.8 *The qubits dropped by $\approx_{delete}$ must not be used in the term, but neither can they be entangled with qubits that are being used. This is expressed in the definition of the equivalence by requiring the relevant state Q_A to be separable.*

2.1.4 The *caseof* statement

We can extend the binary if then else construction to an arbitrary number of bits by defining a *case* statement as a notation, as follows.

Let $\Gamma \vdash b^k : \mathsf{bit}^k$ and $\Delta \vdash t_0 : A, \ldots, \Delta \vdash t_{2^k-1} : A$. Then

$$\mathsf{case}\ b^k\ \mathsf{of}\ \{t_0, \ldots, t_{2^k-1}\} = \begin{cases} t_0 & \text{if } k = 0 \\ \mathsf{let}\ \langle b_0, b^{k-1}\rangle = b^k\ \mathsf{in} & \\ \quad \mathsf{if}\ b_0\ \mathsf{then\ case}\ b^{k-1}\ \mathsf{of}\ \{t_{2^{k-1}}, \ldots, t_{2^k-1}\} & \\ \quad\quad \mathsf{else\ case}\ b^{k-1}\ \mathsf{of}\ \{t_0, \ldots, t_{2^{k-1}-1}\} & \text{if } k > 0 \end{cases}$$

Lemma 2.1.9 *The typing rule for the case construction can be derived using the rules* if *and* $\otimes_E$. *We refer to this derivation as follows:*

$$\frac{\Gamma \vdash b^k : \mathsf{bit}^k \quad \Delta \vdash t_0 : A \quad \ldots \quad \Delta \vdash t_{2^k-1} : A}{\Gamma, \Delta \vdash \mathsf{case}\ b^k\ \mathsf{of}\ \{t_0, \ldots, t_{2^k-1}\} : A}\ \mathsf{case}$$

Proof *We proceed by induction on k.*

- *If $k = 0$. By hypothesis $\Delta \vdash t_0 : A$ and then, by Lemma 2.1.2, $\Gamma, \Delta \vdash t_0 : A = \Gamma, \Delta \vdash$* $\mathsf{case}\ b^0\ \mathsf{of}\ \{t_0\}$.
- *If $k > 1$,*

$$\dfrac{\Gamma \vdash b^k : \mathsf{bit}^k \qquad \dfrac{\dfrac{}{b_0 : \mathsf{bit} \vdash b_0 : \mathsf{bit}}\ \mathsf{ax}_1 \qquad \Pi_1 \qquad \Pi_2}{\Delta, b_0 : \mathsf{bit}, b^{k-1} : \mathsf{bit}^{k-1} \vdash \begin{array}{r}\mathsf{if}\ b_0\ \mathsf{then\ case}\ b^{k-1}\ \mathsf{of}\ \{t_{2^{k-1}}, \ldots, t_{2^k-1}\} \\ \mathsf{else\ case}\ b^{k-1}\ \mathsf{of}\ \{t_0, \ldots, t_{2^{k-1}-1}\}\end{array} : A}\ \mathsf{if}}{\Gamma, \Delta \vdash \begin{array}{r}\mathsf{let}\ \langle b_0, b^{k-1}\rangle = b^k\ \mathsf{in\ if}\ b_0\ \mathsf{then\ case}\ b^{k-1}\ \mathsf{of}\ \{t_{2^{k-1}}, \ldots, t_{2^k-1}\} \\ \mathsf{else\ case}\ b^{k-1}\ \mathsf{of}\ \{t_0, \ldots, t_{2^{k-1}-1}\}\end{array} : A}\ \otimes_E$$

Where

$$\Pi_1 : \quad \dfrac{\dfrac{}{b^{k-1} : \mathsf{bit} \vdash b^{k-1} : \mathsf{bit}}\ \mathsf{ax}_1 \qquad \Delta \vdash t_{2^{k-1}} : A \quad \ldots \quad \Delta \vdash t_{2^k-1} : A}{\Delta, b^{k-1} : \mathsf{bit}^{k-1} \vdash \mathsf{case}\ b^{k-1}\ \mathsf{of}\ \{t_{2^{k-1}}, \ldots, t_{2^k-1}\}}\ \mathsf{case}$$

$$\Pi_2 : \quad \dfrac{\dfrac{}{b^{k-1} : \mathsf{bit} \vdash b^{k-1} : \mathsf{bit}}\ \mathsf{ax}_1 \qquad \Delta \vdash t_0 : A \quad \ldots \quad \Delta \vdash t_{2^{k-1}-1} : A}{\Delta, b^{k-1} : \mathsf{bit}^{k-1} \vdash \mathsf{case}\ b^{k-1}\ \mathsf{of}\ \{t_0, \ldots, t_{2^{k-1}-1}\}}\ \mathsf{case}$$

□

Example 2.1.10 *The following term uses a* case *statement to choose between four variables using a two-bit tuple:*

$$\begin{aligned} & \mathsf{case}\ \langle 0, 1\rangle\ \mathsf{of}\ \{x_0, x_1, x_2, x_3\} \\ = \ & \mathsf{if}\ 0\ \mathsf{then}\ (\mathsf{case}\ 1\ \mathsf{of}\ \{x_2, x_3\})\ \mathsf{else}\ (\mathsf{case}\ 1\ \mathsf{of}\ \{x_0, x_1\}) \\ = \ & \mathsf{if}\ 0\ \mathsf{then}\ (\mathsf{if}\ 1\ \mathsf{then}\ x_3\ \mathsf{else}\ x_2)\ \mathsf{else}\ (\mathsf{if}\ 1\ \mathsf{then}\ x_1\ \mathsf{else}\ x_0) \end{aligned}$$

2.2 λ_ρ, a density matrix calculus

Diaz-Caro introduced λ_ρ in [DC17], a quantum lambda calculus using density matrices to describe quantum data.

The calculus is based on the same semantics as the calculus λ_q presented in Section 2.1, but it encodes the states directly within the terms, without resorting to an external quantum closure. While this makes the programs simpler, it does not allow us to separate entangled qubits as we can do by using pointers in λ_q.

λ_ρ uses a probabilistic reduction system, (cf. Subsection 2.1.2), to model the measurement operation. While mixed state density matrices can be encoded in a term, the probabilistic measurement of a pure state always creates a pure state. In Section 2.3 we present a variation of λ_ρ called λ_ρ° [DC17] that exploits the ability of density matrices to encode all the posible results of a measurement as a mixed state, and generalizes them to arbitrary terms.

λ_ρ extends the simply typed lambda calculus with terms representing the quantum postulates and terms for the classical control, adding a term for measurement results and a branching term based on such results. We refer to the set of terms in λ_ρ as Λ_ρ, and define them as:

$$\begin{aligned} t ::= {} & x \mid tt \mid \lambda x.\, t \mid \\ & \rho^n \mid U^n t \mid \pi^m t \mid t \otimes t \mid \\ & (b^m, \rho^n) \mid \mathsf{letcase}\ x = r\ \mathsf{in}\ \{t, \dots, t\} \end{aligned}$$

Where ρ^n represents an n-qubit density matrix, U^n corresponds to an n-qubit quantum gate applied to the first qubits of the state, π^m represents a measurement of the first m qubits in a state, (b^m, ρ^n) is a pair of an m-bit number representing a measurement result and the resulting density matrix ρ^n, and the $\mathsf{letcase}$ construction chooses between a number of terms based on that result.

The types in λ_ρ are defined as:

$$A ::= n \mid (m, n) \mid A \multimap A$$

Where $n, m \in \mathbb{N}$. The intuition is that a term with type n represents an n-qubit density matrix, (m, n) is the result of a measurement over the first m qubits of an n-qubit state, and $A \multimap B$ corresponds to an affine function. We refer the set of types as Π_ρ.

λ_ρ defines an affine typing system, given in Table 2.3. This typing system admits the weakening rule (Lemma 2.2.1).

$$\frac{}{\Delta, x : A \vdash x : A}\ \mathsf{ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma \vdash t : A \multimap B \quad \Delta \vdash r : A}{\Gamma, \Delta \vdash tr : B}\ \multimap_e$$

$$\frac{}{\Delta \vdash \rho^n : n}\ \mathsf{ax}_\rho \qquad \frac{\Gamma \vdash t : n}{\Gamma \vdash U^m t : n}\ \mathsf{u} \qquad \frac{\Gamma \vdash t : n}{\Gamma \vdash \pi^m t : (m, n)}\ \mathsf{m} \qquad \frac{\Gamma \vdash t : n \quad \Delta \vdash r : m}{\Gamma, \Delta \vdash t \otimes r : n + m}\ \otimes$$

$$\frac{}{\Delta \vdash (b^m, \rho^n) : (m, n)}\ \mathsf{ax_{am}} \qquad \frac{\Delta, x : n \vdash t_0 : A \quad \dots \quad \Delta, x : n \vdash t_{2^m - 1} : A \quad \Gamma \vdash r : (m, n)}{\Gamma, \Delta \vdash \mathsf{letcase}\ x = r\ \mathsf{in}\ t_0, \dots, t_{2^m-1} : A}\ \mathsf{lc}$$

Table 2.3: Typing rules for λ_ρ

We have slightly modified the rule lc from its original presentation in order to allow for an arbitrary context Δ in the typing of each t_i. In a reduction system without a reduction strategy this modification would produce a non confluent calculus [Rom19], but since we will fix an strategy (cf. Section 2.2.1) this does not present a problem in our case.

Lemma 2.2.1 (Weakening) *If $\Gamma \vdash t : A$ and $x \notin \mathsf{FV}(t)$, then $\Gamma, x : B \vdash t : A$.* $\square$

Example 2.2.2 *We can rewrite the fair coin example from Example 2.1.1 in λ_ρ. Let t and r be two arbitrary terms and let H be the Hadamard gate, we can describe the choice between the two terms by flipping a fair coin as:*

$$\mathsf{letcase}\ x = \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|)\ \mathsf{in}\ \{t, r\}$$

Notice that we could also have written $\mathsf{H}\ \ |0\rangle\langle 0|$ directly as the density matrix $|+\rangle\langle +|$.

Remark 2.2.3 *When applying unitary operators to density matrices, we use $U\rho$ as syntactic notation for $U\rho U^\dagger$.*

Example 2.2.4 *Given a context $\Gamma = t : A,\ r : A$, we can write the following type derivation for the term in Example 2.2.2:*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash |0\rangle\langle 0| : 1}\ \mathsf{ax}_\rho}{\vdash \mathsf{H}\ \ |0\rangle\langle 0| : 1}\ \mathsf{u}}{\vdash \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|) : (1,1)}\ \mathsf{m} \qquad \dfrac{}{\Gamma, x : 1 \vdash t : A}\ \mathsf{ax} \qquad \dfrac{}{\Gamma, x : 1 \vdash r : A}\ \mathsf{ax}}{\Gamma \vdash \mathsf{letcase}\ x = \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|)\ \mathsf{in}\ \{t, r\} : A}\ \mathsf{if}$$

2.2.1 Rewrite system and reduction strategy

λ_ρ has no defined strategy, but since λ_q's reduction (cf. Subsection 2.1.2) is call-by-value, we fix the same strategy. We could also have fixed a call-by-name strategy and used the technique described in [Plo75] to simulate call-by-value.

λ_ρ has the affine probabilistic reduction system given in Table 2.4. If U^m is applied to a state ρ^n, with $m \leq n$, we write $\overline{U^m}$ the extension of the gate U^m to the higher dimension using the identity operator, $\overline{U^m} = U^m \otimes I^{n-m}$. Similarly, we extend the m-qubit observables $\{\pi_i^m\}_i$ of a measurement π^m to measure the only first qubits of an n-qubit system as $\overline{\pi_i^m} = \pi_i^m \otimes I^{n-m}$.

The presentation of λ_ρ given here is slightly modified from its original version to consider $\rho_1 \otimes \rho_2 = \rho_3$ instead of $\rho_1 \otimes \rho_2 \longrightarrow_1 \rho_3$. Indeed, there is no computational meaning for such a rewrite rule. In Section 3.1 we discuss the translation in the presence of this rule.

We write the set of values of λ_ρ as V_ρ. They are defined as follows:

$$v := x \mid \lambda x.t \mid \rho^n \mid (b^m, \rho^n)$$

Example 2.2.5 *The following trace corresponds to the coin-flipping term from Example 2.2.2.*

$$
\begin{array}{rcll}
(\lambda x.\ t)V & \longrightarrow_1 & tr[V/x] & \\
U^m \rho^n & \longrightarrow_1 & \rho'^n & \text{with } \rho'^n = \overline{U^m}\,\rho^n\,\overline{U^m}^\dagger \\
\pi^m \rho^n & \longrightarrow_{p_i} & (i, \rho_i^n) & \text{with } \begin{cases} p_i = \mathrm{tr}(\overline{\pi_i^m}^\dagger\,\overline{\pi_i^m}\,\rho^n) \\ \rho_i^n = \frac{\overline{\pi_i^m}^\dagger\,\rho^n\,\overline{\pi_i^m}}{p_i} \end{cases} \\
\mathsf{letcase}\ x = (b^m, \rho^n)\ \mathsf{in}\ \{t_0, \dots, t_{2^m-1}\} & \longrightarrow_1 & t_{b^m}[\rho^n/x] & \\
\rho_1^n \otimes \rho_2^m & = & \rho^{n+m} & \text{with } \rho^{n+m} = \rho_1^n \otimes \rho_2^m
\end{array}
$$

$$
\frac{t \longrightarrow_p r}{t\ v \longrightarrow_p r\ v} \qquad \frac{t \longrightarrow_p r}{s\ t \longrightarrow_p s\ r} \qquad \frac{t \longrightarrow_p r}{U^n t \longrightarrow_p U^n r} \qquad \frac{t \longrightarrow_p r}{\pi^n t \longrightarrow_p \pi^n r}
$$

$$
\frac{t \longrightarrow_p r}{\mathsf{letcase}\ x = t\ \mathsf{in}\ \{s_0, \dots, s_n\} \longrightarrow_p \mathsf{letcase}\ x = r\ \mathsf{in}\ \{s_0, \dots, s_n\}}
$$

Table 2.4: Rewrite system for λ_ρ

$$
\begin{tikzcd}[column sep=tiny]
 & \mathsf{letcase}\ x = \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|)\ \mathsf{in}\ \{t, r\} \arrow[d, "1"] & \\
 & \mathsf{letcase}\ x = \pi^1\ |+\rangle\langle +|\ \ \mathsf{in}\ \{t, r\} \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & \\
\mathsf{letcase}\ x = (0, |0\rangle\langle 0|)\ \mathsf{in}\ \{t, r\} \arrow[d, "1"] & & \mathsf{letcase}\ x = (1, |1\rangle\langle 1|)\ \mathsf{in}\ \{t, r\} \arrow[d, "1"] \\
t & & r
\end{tikzcd}
$$

2.2.2 Denotational semantics

We use the interpretation of typed terms into generalized mixed states for λ_ρ described in [DC17].

Types are interpreted into sets of density matrices and functions. We define it in Table 2.5. $\mathcal{D}_n$ is the set of n-qubit density matrices, trd is a helper function such that $\mathsf{trd}(\{(p_i, b_i, e_i)\}_i) = \{e_i\}_i$, w is a weight function $\mathsf{w}(\{(p_i, b_i, e_i)\}_i) = \sum_i p_i$, and $P(b, A)$ is the following proposition: $[(A = \vec{B} \multimap (m, n)) \Rightarrow b \neq \varepsilon]$.

$$
\begin{aligned}
[\![n]\!] &= \mathcal{D}_n \\
[\![(m, n)]\!] &= \mathcal{D}_n \\
[\![A \multimap B]\!] &= \{f \mid \forall e \in [\![A]\!], \forall b \in \mathbb{N}^\varepsilon \text{ s.t. } P(b, A), \\
&\qquad \mathsf{trd}(f(b, e)) \subseteq [\![B]\!], \mathsf{w}(f(b, e)) = 1 \text{ and } P(f(b, e), B)\}
\end{aligned}
$$

Table 2.5: Type interpretation for λ_ρ

Let $\mathbb{N}^\varepsilon = \mathbb{N}_0 \cup \{\varepsilon\}$. Terms are interpreted into sets of tuples (p, b, e) where $p \in (0, 1]$

represents a probability, $b \in \mathbb{N}^\varepsilon$ is the output of a measurement if it occurred, and $e \in [\![A]\!]$ for some type A. In Table 2.6 we define the interpretation with respect to a valuation $\theta : \mathcal{V} \to \mathbb{N}^\varepsilon \times E$, where $E = \bigcup_{A \in \mathsf{Types}} [\![A]\!]$.

$$
\begin{array}{rll}
[\![x]\!]_\theta &= \{(1, b, e)\} & \text{where } \theta(x) = (b, e) \\
[\![\lambda x.t]\!]_\theta &= \{(1, \varepsilon, (b, e) \mapsto [\![t]\!]_{\theta, x=(b,e)})\} & \\
[\![t_1\, t_2]\!]_\theta &= \{(p_i\, q_j\, h_{ijk}, b''_{ijk}, g_{ijk}) \mid [\![r]\!]_\theta = \{(p_i, b_i, e_i)\}_i, & \\
& \qquad [\![t]\!]_\theta = \{(q_j, b'_j, f_i)\}_j, \text{and} & \\
& \qquad f_j(b_i, e_i) = \{(h_{ijk}, b''_{ijk}, g_{ijk})\}_k\} & \\
[\![\rho^n]\!]_\theta &= \{(1, \varepsilon, \rho^n)\} & \\
[\![U^n t]\!]_\theta &= \{(p_i, \varepsilon, \overline{U^n} \rho_i \overline{U^n}^\dagger) \mid [\![t]\!]_\theta = \{(p_i, b_i, \rho_i)\}_j\} & \\
[\![\pi^m t]\!]_\theta &= \{(p_j\, \mathsf{tr}(\overline{\pi_i}^\dagger \overline{\pi_i} \rho_j), i, \dfrac{\overline{\pi_i} \rho_j \overline{\pi_i}^\dagger}{\mathsf{tr}(\overline{\pi_i}^\dagger \overline{\pi_i} \rho_j)}) \mid [\![t]\!]_\theta = \{(p_j, b_j, \rho_j)\}_j\} & \\
[\![t \otimes r]\!]_\theta &= \{(p_i\, q_j, \varepsilon, \rho_i \otimes \rho'_j) \mid [\![t]\!]_\theta = \{(p_i, b_i, \rho_i)\}_i, [\![r]\!]_\theta = \{(q_j, b'_j, \rho'_j)\}_j\} & \\
[\![(b^m, \rho^n)]\!]_\theta &= \{(1, b^m, \rho^n)\} & \\
[\![\mathsf{letcase}\ x = r\ \mathsf{in}\ \{t_0, \dots, t_{2^m-1}\}]\!]_\theta &= \{(p_i\, q_{ij}, b'_{ij}, e_{ij}) \mid [\![r]\!]_\theta = \{(p_i, b_i, \rho_i)\}_i, \text{and} & \\
& \qquad [\![t_{b_i}]\!]_{\theta, x=(\varepsilon, \rho_i)} = \{(q_{ij}, b'_{ij}, e_{ij})\}_j\} &
\end{array}
$$

Table 2.6: Term interpretation for λ_ρ

Example 2.2.6 *We can determine the interpretation corresponding to the coin-flipping term from Example 2.2.2 under a valuation θ. We have*

$$[\![\mathsf{H}\ \ |0\rangle\langle 0|]\!]_\theta = \{(1, \varepsilon, |+\rangle\langle +|)\}$$

$$[\![\pi^1(\mathsf{H}\ \ |0\rangle\langle 0|)]\!]_\theta = \{(\frac{1}{2}, 0, |0\rangle\langle 0|), (\frac{1}{2}, 1, |1\rangle\langle 1|)\}.$$

Let $[\![t]\!]_\theta = \{(1, b_t, e_t)\}$ *and* $[\![r]\!]_\theta = \{(1, b_r, e_r)\}$ *where* $\theta(t) = (b_t, e_t)$ *and* $\theta(t) = (b_r, e_r)$ *hence,*

$$[\![\mathsf{letcase}\ x = \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|)\ \mathsf{in}\ \{t, r\}]\!]_\theta = \{(\frac{1}{2}, b_t, e_t), (\frac{1}{2}, b_r, e_r)\}.$$

2.3 λ_ρ^o, the generalized λ_ρ

In [DC17], Diaz-Caro also defines a variation to the λ_ρ calculus presented in Section 2.2 called λ_ρ^o, or *generalized lambda rho*. This calculus replaces the probabilistic reduction system with a deterministic system in which a $\mathsf{letcase}$ on a measurement operation reduces to a linear combination of the resulting terms,

$$\mathsf{letcase}^o\ x = \pi^m \rho^n\ \mathsf{in}\ t_0, \dots, t_{2^m} - 1 \rightsquigarrow \sum_i p_i t_i[\rho_i^n / x].$$

We call this linear combination of terms a *generalized density matrix*, and define it following the grammar of the algebraic lambda-calculi [AD17, ADCP$^+$14, Vau09].

Terms in λ_ρ^o correspond to the terms in λ_ρ, replacing the measurement result with the linear combination of terms. We refer to the set of terms as Λ_ρ^o.

$$t ::= x \mid tt \mid \lambda x.t \mid \rho^n \mid U^n t \mid \pi^n t \mid t \otimes r \mid \sum_{i=1}^{n} p_i t_i \mid \mathsf{letcase}^o\ x = r \text{ in } \{t, \dots, t\}$$

Where $p_i \in (0,1]$, $\sum_{i=1}^{n} p_i = 1$, and $\sum$ is considered modulo associativity and commutativity.

λ_ρ^o uses the same set of types as λ_ρ, Π_ρ, defined as:

$$\textit{type } A ::= n \mid (m,n) \mid A \multimap A$$

The type system for λ_ρ^o is defined in Table 2.7. As in λ_ρ, we allow for an arbitrary context Δ in the rule lc^o.

$$\frac{}{\Delta, x : A \vdash x : A}\ \mathsf{ax} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma \vdash t : A \multimap B \quad \Delta \vdash r : A}{\Gamma, \Delta \vdash tr : B}\ \multimap_e$$

$$\frac{}{\Delta \vdash \rho^n : n}\ \mathsf{ax}_\rho \qquad \frac{\Gamma \vdash t : n}{\Gamma \vdash U^m t : n}\ \mathsf{u} \qquad \frac{\Gamma \vdash t : n}{\Gamma \vdash \pi^m t : (m,n)}\ \mathsf{m} \qquad \frac{\Gamma \vdash t : n \quad \Delta \vdash r : m}{\Gamma, \Delta \vdash t \otimes r : n + m}\ \otimes$$

$$\frac{\Delta, x : n \vdash t_0 : A \quad \dots \quad \Delta, x : n \vdash t_{2^m - 1} : A \quad \Gamma \vdash r : (m,n)}{\Gamma, \Delta \vdash \mathsf{letcase}^o\ x = r \text{ in } t_0, \dots, t_{2^m-1} : A}\ \mathsf{lc}^o$$

$$\frac{\Gamma \vdash t_1 : A \quad \dots \quad \Gamma \vdash t_n : A \quad \sum_{i=1}^{n} p_i = 1}{\Gamma \vdash \sum_{i=1}^{n} p_i t_i : A}\ +$$

Table 2.7: Typing rules for λ_ρ^o

Example 2.3.1 *The fair coin example from Example 2.1.1 can be written in λ_ρ^o with the same term from λ_ρ as in Example 2.2.2,*

$$\mathsf{letcase}^o\ x = \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|) \text{ in } \{t, r\},$$

but in this case we can also write the linear combination between t and r directly as

$$\frac{1}{2}t + \frac{1}{2}r.$$

Example 2.3.2 *Given a context Γ such that $\Gamma \vdash t : A$ and $\Gamma \vdash r : A$, we can write a type derivation for the sum term in Example 2.3.1 using the rule $+$ as follows:*

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash r : A}{\Gamma \vdash \frac{1}{2}t + \frac{1}{2}r : A}\ +$$

2.3.1 Rewrite system

As discussed for λ_ρ in Section 2.2.1, we choose to use a weak call-by-value strategy for λ_ρ^o. The rewrite rules are defined in table 2.8. As in Section 2.2.1, we write $\overline{U^m}$ and $\overline{\pi_i^m}$ for the extension to n-qubit states of m-qubit operators and observables.

We refer to the set of values of λ_ρ^o as V_ρ^o. They are defined as:

$$v := x \mid \rho^n \mid \lambda x.\ t \mid \sum_i p_i v_i \text{ with } v_i \neq v_j \text{ if } i \neq j$$

$$\begin{array}{rcll}
(\lambda x.t)r & \rightsquigarrow & t[r/x] & \\
\mathsf{letcase}^o\ x = \pi^m \rho^n \text{ in } t_0, \dots, t_{2^m-1} & \rightsquigarrow & \sum_i p_i t_i[\rho_i^n/x] & \text{with } \begin{cases} \rho_i^n = \frac{\overline{\pi_i^m}\rho^n\overline{\pi_i^m}^\dagger}{p_i} \\ p_i = \mathsf{tr}(\overline{\pi_i^m}^\dagger\overline{\pi_i^m}\rho^n) \end{cases} \\
U^m \rho^n & \rightsquigarrow & \rho'^n & \text{with } \overline{U^m}\rho^n\overline{U^m}^\dagger = \rho'^n \\
\sum_i p_i \rho_i & \rightsquigarrow & \rho' & \text{with } \rho' = \sum_i p_i\rho_i \\
\sum_i p_i t & \rightsquigarrow & t & \\
(\sum_i p_i t_i) r & \rightsquigarrow & \sum_i p_i(t_i r) & \\
\rho_1^n \otimes \rho_2^m & = & \rho^{n+m} & \text{with } \rho^{n+m} = \rho_1^n \otimes \rho_2^m
\end{array}$$

$$\frac{t \rightsquigarrow r}{tv \rightsquigarrow rv} \qquad \frac{t \rightsquigarrow r}{st \rightsquigarrow sr} \qquad \frac{t \rightsquigarrow r}{U^n t \rightsquigarrow U^n r}$$

$$\frac{t_j \rightsquigarrow r_j}{\sum_{i=1}^n p_i t_i \rightsquigarrow \sum_{i=1}^n p_i r_i}\ {\scriptstyle (\forall i \neq j, t_i = r_j)}$$

$$\frac{t \rightsquigarrow r}{\mathsf{letcase}^o\ x = t \text{ in } s_0, \dots, s_{2^m-1} \rightsquigarrow \mathsf{letcase}^o\ x = r \text{ in } s_0, \dots, s_{2^m-1}}$$

Table 2.8: Rewrite system for λ_ρ^o

Example 2.3.3 *The following deterministic trace corresponds to the coin-flipping term in Example 2.3.1:*

$$\mathsf{letcase}^o\ x = \pi^1(\mathsf{H}\ \ |0\rangle\langle 0|) \text{ in } \{t, r\} \rightsquigarrow \mathsf{letcase}^o\ x = \pi^1\ |+\rangle\langle +|\ \text{ in } \{t, r\} \rightsquigarrow \frac{1}{2}t + \frac{1}{2}r$$

2.3.2 Interpretation

As in [DC17] we use the interpretation for types introduced in 2.2.2 for λ_ρ. It suffices to make a small modification to the interpretation of terms, dropping the term (b^m, ρ^n) and defining an interpretation for the sum term as follows:

$$[\![\sum_i p_i\, t_i]\!]_\theta = \{(p_i\, q_{ij}, b_{ij}, e_{ij}) \mid [\![t_i]\!]_\theta = \{(q_{ij}, b_{ij}, e_{ij})\}_j\}$$

Example 2.3.4 *Given the terms $t, r \in \Lambda_\rho^o$ and a valuation θ such that $[\![t]\!]_\theta = \{(1, b_t, e_t)\}$ and $[\![r]\!]_\theta = \{(1, b_r, e_r)\}$, we can calculate the type interpretation for the linear term in the coin-flipping Example 2.3.1 as follows:*

$$[\![\frac{1}{2}\,t + \frac{1}{2}\,r]\!]_\theta = \{(\frac{1}{2}, b_t, e_t), (\frac{1}{2}, b_r, e_r)\}.$$

Notice that this is the same interpretation as for the equivalent coin-flipping term in Example 2.3.4. This is not casual. Both calculi have the same interpretation, and hence can be considered as two representations of the same phenomenon.

Chapter 3

Simulations

In this chapter we define a number of translations between the calculi λ_ρ, λ_ρ^o, and λ_q following the schema bellow, where the dashed lines represent inverse translations defined over the image of the forward translation.

$$\begin{tikzcd} \lambda_\rho^o \arrow[r, bend left, "\{\!\{\cdot\}\!\}"] & \lambda_\rho \arrow[l, bend left, dashed, "\{\!\{\cdot\}\!\}^{-1}"] \arrow[r, bend left, "(\!|\cdot|\!)"] & \lambda_q \arrow[l, bend left, dashed, "(\!|\cdot|\!)^{-1}"] \end{tikzcd}$$

In Section 3.1 we define $(\!|\cdot|\!)$, a translation from λ_ρ to λ_q. In Section 3.2 we then define $(\!|\cdot|\!)^{-1}$, a left inverse for the translation $(\!|\cdot|\!)$. In Section 3.3, we define a translation between λ_ρ^o and λ_ρ denoted as $\{\!\{\cdot\}\!\}$ that we compose with $(\!|\cdot|\!)$ to obtain a full translation from λ_ρ^o to λ_q. And finally, in Section 3.4 we define $\{\!\{\cdot\}\!\}^{-1}$, a pseudoinverse for $\{\!\{\cdot\}\!\}$.

3.1 Translation from λ_ρ to λ_q

In this section we define the translation $(\!|\cdot|\!)$ from terms in λ_ρ to closures in λ_q.

Terms in λ_ρ may contain mixed state density matrices, which cannot be directly translated to a quantum state in λ_q, since quantum closures can only describe pure states. Hence, we first define a purification method for density matrices.

3.1.1 Mixed state purification

Based on the state purification technique (cf. [NC10, Chapter I-Section 2.5]), we define a purification function for mixed state density matrices. The idea of the method is to add extra qubits to the system to encode the mixed states as part of a bigger pure states.

Suppose that we are given a state ρ^A of an n-qubit quantum system A. Since ρ^A is a positive semidefinite matrix, it has a spectral decomposition $\rho^A = \sum_{i=1}^{2^n} \lambda_i \, |v_i\rangle\langle v_i|$, where $\{|v_i\rangle\}$ is an orthonormal basis of eigenvectors [KC09].

Let B be another n-qubit quantum system and let $\{\,|e_i\rangle\,\}_{i=1,\dots,n}$ be an ortonormal basis for B, such that if ρ^A describes a mixed state, the following purification of ρ^A is a pure state in the system $A \otimes B$:

$$\mathsf{pur}(\rho^A) = |\psi\rangle\langle\psi|\,, \text{ where } |\psi\rangle = \sum_i \sqrt{\lambda_i}\,|v_i\rangle \otimes |e_i\rangle\,.$$

If ρ^A already corresponds to a pure state, then $\mathsf{pur}(\rho^A) = \rho^A$.

Notice that ρ^A corresponds to the reduced density operator of $\mathsf{pur}(\rho^A)$ for the system A described in Section 1.2,

$$\rho^A = \mathsf{tr}_B(\mathsf{pur}(\rho^A)) \text{ where } \mathsf{tr}_\emptyset = \mathsf{id}.$$

Example 3.1.1 *Let ρ describe the fully mixed state of a one qubit quantum system:*

$$\rho = \frac{1}{2}\,|0\rangle\langle 0| + \frac{1}{2}\,|1\rangle\langle 1| = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

Let B be another one-qubit system and let us choose the orthonormal basis for B, $\{|0\rangle\,, |1\rangle\}$ in the purification. We can find a pure state that contains ρ by purifying it through B:

$$\mathsf{pur}(\rho) = |\psi\rangle\langle\psi|\,, \textit{ where } |\psi\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle\,,$$

$$\mathsf{pur}(\rho) = \frac{1}{2}\,|00\rangle\langle 00| + \frac{1}{2}\,|00\rangle\langle 11| + \frac{1}{2}\,|11\rangle\langle 00| + \frac{1}{2}\,|11\rangle\langle 11| = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

Notice that we can get ρ back from the purified state:

$$\mathsf{tr}_B(\mathsf{pur}(\rho)) = \frac{1}{2}\,|0\rangle\langle 0| + \frac{1}{2}\,|1\rangle\langle 1| = \rho$$

3.1.2 Translation definition

In λ_ρ the quantum state is described directly in the terms as density matrices. Instead, λ_q uses a quantum closure containing a single external quantum state, and uses *pointer* variables in the term to refer to specific qubits. We must, therefore, translate λ_ρ-terms to quantum closures in λ_q in order to encode the density matrices.

We define the translation $(\!|\cdot|\!) : \Lambda_\rho \to \mathcal{C}_q$ inductively in Table 3.1.

Since both calculi include the simply typed lambda calculus, we translate it directly and combine any quantum states from the translated subterms. We also define the translations for the unitary operator term, the tensor term, and the measurement result in the same manner.

The translation of density matrices makes use of the purification method defined previously. We encode any mixed state density matrix as a bigger pure state, and reference only the first qubits that correspond to the original quantum system.

For the measurement term π^m we must translate an m-qubit measurement from λ_ρ using the single qubit meas operation from λ_q, and return a tuple that contains both the result and the collapsed state, mimicking the operation performed by λ_ρ. Since we cannot duplicate qubit variables, we are forced to recreate the measured qubits using new operations and forget about the measured qubits. Since we consider quantum closures

$$
\begin{aligned}
(\!|x|\!) &= [*, \varnothing, x] \\
(\!|\lambda x.\ t|\!) &= [\mathrm{st}((\!|t|\!)), \mathrm{lnk}((\!|t|\!)),\ \lambda x.\ \mathrm{term}((\!|t|\!))] \\
(\!|t_1\ t_2|\!) &= [\mathrm{st}((\!|t_1|\!)) \otimes \mathrm{st}((\!|t_2|\!)), \mathrm{lnk}((\!|t_1|\!)) \cup \mathrm{lnk}((\!|t_2|\!)), \mathrm{term}((\!|t_1|\!))\ \mathrm{term}((\!|t_2|\!))] \\
(\!|\rho^n|\!) &= [|\phi\rangle, \{x_i \mapsto i\}_{i=1}^n, \langle x_1, \ldots, x_n\rangle] \qquad \text{where } \mathsf{pur}(\rho^n) = |\phi\rangle\!\langle\phi| \\
(\!|U^n\ t|\!) &= [\mathrm{st}((\!|t|\!)), \mathrm{lnk}((\!|t|\!)),\ U^n\, \mathrm{term}((\!|t|\!))] \\
(\!|\pi^m\ t|\!) &= [\mathrm{st}((\!|t|\!)), \mathrm{lnk}((\!|t|\!)), \mathsf{let}\ \langle x_1,\ \ldots,\ x_n\rangle = \mathrm{term}((\!|t|\!)) \\
&\qquad \mathsf{in\ let}\ \langle b_1,\ \ldots,\ b_m\rangle = \langle \mathsf{meas}\ x_1,\ \ldots,\ \mathsf{meas}\ x_m\rangle \\
&\qquad\quad \mathsf{in}\ \langle b_1,\ \ldots,\ b_m,\ \mathsf{new}\ b_1,\ \ldots,\ \mathsf{new}\ b_m,\ x_{m+1},\ \ldots,\ x_n\rangle] \\
(\!|t_1 \otimes t_2|\!) &= [\mathrm{st}((\!|t_1|\!)) \otimes \mathrm{st}((\!|t_2|\!)), \mathrm{lnk}((\!|t_1|\!)) \cup \mathrm{lnk}((\!|t_2|\!)), \langle \mathrm{term}((\!|t_1|\!)),\ \mathrm{term}((\!|t_2|\!))\rangle] \\
(\!|b^m, \rho^n|\!) &= [\mathrm{st}((\!|\rho^n|\!)), \mathrm{lnk}((\!|\rho^n|\!)),\ \langle b, \mathrm{term}((\!|\rho^n|\!))\rangle] \\
&\qquad \text{where } b \text{ is } b^m \text{ expressed as a m-uple of } \{0,1\} \\
(\!|\mathsf{letcase}\ x = r\ \mathsf{in}\ \{t_1, \ldots, t_{2^n}\}|\!) & \\
&= [\mathrm{st}((\!|r|\!)) \otimes \bigotimes_{i=1}^{2^n} \mathrm{st}((\!|t_i|\!)), \mathrm{lnk}((\!|r|\!)) \cup \bigcup_{i=1}^{2^n} \mathrm{lnk}((\!|t_i|\!)), \\
&\quad \mathsf{let}\ \langle b, x\rangle = \mathrm{term}((\!|r|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{\mathrm{term}((\!|t|\!)_1), \ldots, \mathrm{term}((\!|t_{2^n}|\!))\}]
\end{aligned}
$$

Table 3.1: Translation from λ_ρ to λ_q

modulo $\approx$-equivalence (cf. Subsection 2.1.3) we can discard the leftover qubits from the state.

The $\mathsf{letcase}$ translation is defined as a tree of $\mathsf{if\ then\ else}$, using the caseof statement defined in Section 2.1.

Example 3.1.2 *Consider the coin-flipping term in Example 2.2.2,*

$$t = \mathsf{letcase}\ x = \pi^1(\mathsf{H}\ \ |0\rangle\!\langle 0|)\ \mathsf{in}\ \{r_1, r_2\}.$$

The translation of t results in a similar term in λ_q, using the corresponding operations:

$$
\begin{aligned}
(\!|t|\!) = {} & [st((\!|\pi^1(\mathsf{H}\ \ |0\rangle\!\langle 0|)|\!)) \otimes st((\!|r_1|\!)) \otimes st((\!|r_2|\!)), lnk((\!|\pi^1(\mathsf{H}\ \ |0\rangle\!\langle 0|)|\!)) \cup lnk((\!|r_1|\!)) \cup lnk((\!|r_2|\!)), \\
& \mathsf{let}\ \langle b, x\rangle = term((\!|\pi^1(\mathsf{H}\ \ |0\rangle\!\langle 0|)|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{(\!|r_1|\!), (\!|r_2|\!)\}] \\
= {} & [|0\rangle, \{x_0 \mapsto 0\}, \\
& \mathsf{let}\ \langle b, x\rangle = (\mathsf{let}\ \langle y_0\rangle = \mathsf{H}\ \langle x_0\rangle\ \mathsf{in\ let}\ \langle b_0\rangle = \langle \mathsf{meas}\ y_0\rangle\ \mathsf{in}\ \langle b_0, new\ b_0\rangle) \\
& \mathsf{in\ case}\ b\ \mathsf{of}\ \{r_1, r_2\}]
\end{aligned}
$$

Example 3.1.3 *Consider the term that measures the first qubit in a two-qubit state and matches on the result,*

$$t = \mathsf{letcase}\ x = \pi^1\ |{+0}\rangle\!\langle {+0}|\ \ \mathsf{in}\ \{x, x\}.$$

The translation of t has to assign variables to each qubit in the pair $(\!|\,|{+0}\rangle\!\langle{+0}|\,|\!)$ *to* *measure only the first one:*

$$\begin{aligned}(\!|t|\!) &= [st((\!|\pi^1 \,|{+0}\rangle\!\langle{+0}|\,|\!)), lnk((\!|\pi^1 \,|{+0}\rangle\!\langle{+0}|\,|\!)),\\ &\quad \mathsf{let}\ \langle b, x\rangle = term((\!|\pi^1 \,|{+0}\rangle\!\langle{+0}|\,|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{x, x\}]\\ &= [|{+0}\rangle, \{x_0 \mapsto 0, x_1 \mapsto 1\},\\ &\quad \mathsf{let}\ \langle b, x\rangle = (\mathsf{let}\ \langle y_0, y_1\rangle = \langle x_0, x_1\rangle\ \mathsf{in}\\ &\qquad \mathsf{let}\ \langle b_0\rangle = \langle \mathsf{meas}\ y_0\rangle\ \mathsf{in}\ \langle b_0, new\ b_0,\ y_1\rangle)\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{x, x\}]\end{aligned}$$

Types and contexts are translated as shown in Table 3.2.

$$(\!|\cdot|\!)\ :\ \Pi_\rho\ \rightarrow\ \Pi_q$$

$$\begin{aligned}(\!|n|\!) &= \mathsf{qbit}^n\\ (\!|(m, n)|\!) &= \mathsf{bit}^m \otimes \mathsf{qbit}^n\\ (\!|A \multimap B|\!) &= (\!|A|\!) \multimap (\!|B|\!)\\[1em] (\!|\Gamma|\!) &= \{x : (\!|A|\!) \mid x : A \in \Gamma\}\end{aligned}$$

Table 3.2: Type translation from λ_ρ to λ_q

Let $c \in \mathcal{C}_q$ (the set of quantum closures) and $t = \text{term}(t)$, then we write $\mathsf{FV}_\Gamma(t)$ the set of free variables in t that are not in the typing context Γ. Furthermore, we write $\mathsf{FV}_\Gamma(t) : \mathsf{qbit}$ the context in λ_q composed by the typing judgement $x : \mathsf{qbit}$ for each variable in the set.

$$\mathsf{FV}_\Gamma(c) : \mathsf{qbit} = \{x : \mathsf{qbit} \mid x \in \mathsf{FV}_\Gamma(c)\}$$

If $\Gamma \vdash c$, then $\mathsf{FV}_\Gamma(t)$ corresponds to the variables in t used as pointers to qubits in the quantum register, and $\Gamma, \mathsf{FV}_\Gamma(t) : \mathsf{qbit} \vdash t$.

Example 3.1.4 (Context translation) *Let Γ be the following typing context in λ_ρ:*

$$\Gamma = \{x : 2, y : 2 \multimap (1, 2)\}$$

The context translation translates each type separately:

$$(\!|\Gamma|\!) = \{x : \mathsf{qbit}^2, y : \mathsf{qbit}^2 \multimap \mathsf{bit} \otimes \mathsf{qbit}^2\}$$

Example 3.1.5 (Free variables) *Consider a quantum closure*

$$c = [|\phi\rangle, \{y \mapsto 0\}, \mathsf{if}\ x\ \mathsf{then\ new}\ 0\ \mathsf{else}\ y]$$

and a typing context $\Gamma = \{x : \mathsf{bit}\}$. Then, $\mathsf{FV}_\Gamma(c) = \{y\}$ and $\mathsf{FV}_\Gamma(c) : \mathsf{qbit} = \{y : \mathsf{qbit}\}$.

3.1.3 Correctness

We prove below that when a well-typed term is translated, the type of the resulting closure matches that of the translated type.

Theorem 3.1.6 *If $t \in \Lambda_\rho$ and $\Gamma \vdash t : A$, then $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis A\rrparenthesis$*

Proof *We proceed by induction on the derivation of $\Gamma \vdash t : A$.*

- *Let $\Gamma, x : A \vdash x : A$ as a consequence of rule* ax*. By rule* ax_1*, we have $\llparenthesis\Gamma\rrparenthesis, x : \llparenthesis A\rrparenthesis \vdash x : \llparenthesis A\rrparenthesis$.*

  Notice that $\mathsf{FV}_{\llparenthesis\Gamma, x:A\rrparenthesis}(term(\llparenthesis x\rrparenthesis)) = \emptyset$, $\llparenthesis\Gamma, x : A\rrparenthesis = \llparenthesis\Gamma\rrparenthesis, x : \llparenthesis A\rrparenthesis$, and $term(\llparenthesis x\rrparenthesis) = x$.

- *Let $\Gamma \vdash \lambda x.t : A \multimap B$ as a consequence of $\Gamma, x : A \vdash t : B$ and rule $\multimap_i$. By the induction hypothesis, $\llparenthesis\Gamma, x : A\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma,x:A\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis B\rrparenthesis$. That is, $\llparenthesis\Gamma\rrparenthesis, x : \llparenthesis A\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma,x:A\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis b\rrparenthesis$. Then, by rule $\multimap_I$, we have $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(\lambda x.term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash \lambda x.term(\llparenthesis t\rrparenthesis) : \llparenthesis A\rrparenthesis \multimap \llparenthesis B\rrparenthesis$.*

  Notice that $\lambda x.term(\llparenthesis t\rrparenthesis) = term(\llparenthesis\lambda x.t\rrparenthesis)$, $\llparenthesis A\rrparenthesis \multimap \llparenthesis B\rrparenthesis = \llparenthesis A \multimap B\rrparenthesis$, and $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(\lambda x.term(\llparenthesis t\rrparenthesis)) = \mathsf{FV}_{\llparenthesis\Gamma,x:A\rrparenthesis}(term(\llparenthesis t\rrparenthesis))$.

- *Let $\Gamma, \Delta \vdash tr : B$ as a consequence of $\Gamma \vdash t : A \multimap B$, $\Delta \vdash r : A$ and rule $\multimap_e$. By the induction hypothesis, $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis A \multimap B\rrparenthesis$, that is $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis A\rrparenthesis \multimap \llparenthesis B\rrparenthesis$, and $\llparenthesis\Delta\rrparenthesis, \mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis r\rrparenthesis) : \llparenthesis A\rrparenthesis$. By rule $\multimap_E$, we have $\llparenthesis\Gamma\rrparenthesis, \llparenthesis\Delta\rrparenthesis$, $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit}$, $\mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis)term(\llparenthesis r\rrparenthesis) : \llparenthesis B\rrparenthesis$.*

  Notice that $\llparenthesis\Gamma, \Delta\rrparenthesis = \llparenthesis\Gamma\rrparenthesis, \llparenthesis\Delta\rrparenthesis$, $term(\llparenthesis tr\rrparenthesis) = term(\llparenthesis t\rrparenthesis)term(\llparenthesis r\rrparenthesis)$, and $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis,\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis tr\rrparenthesis)) = \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)), \mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis))$.

- *Let $\Gamma \vdash \rho^n : n$ as a consequence of rule ax_ρ. By rules ax_1 and $\otimes_I$, we have $\llparenthesis\Gamma\rrparenthesis$, $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(\langle x_1, \dots, x_n\rangle) : \mathsf{qbit} \vdash \langle x_1, \dots, x_n\rangle : \mathsf{qbit}^n$.*

  Notice that $term(\llparenthesis\rho^n\rrparenthesis) = \langle x_1, \dots, x_n\rangle$, $\llparenthesis n\rrparenthesis = \mathsf{qbit}^n$, and since the x_i are fresh variables, $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(\langle x_1, \dots, x_n\rangle) : \mathsf{qbit} = \{x_1 : \mathsf{qbit}, \dots, x_n : \mathsf{qbit}\}$.

- *Let $\Gamma \vdash U^n t : n$ as a consequence of $\Gamma \vdash t : n$ and rule* u*. By the induction hypothesis, $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis n\rrparenthesis$. That is, $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(U^n term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \mathsf{qbit}^n$. By rule ax_2, we have $\vdash U^n : \mathsf{qbit}^n \multimap \mathsf{qbit}^n$. Then, by rule $\multimap_E$, we have $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(U^n term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash U^n term(\llparenthesis t\rrparenthesis) : \mathsf{qbit}^n$.*

  Notice that $U^n\, term(\llparenthesis t\rrparenthesis) = term(\llparenthesis U^n t\rrparenthesis)$, $\llparenthesis n\rrparenthesis = \mathsf{qbit}^n$, and $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) = \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis U^n t\rrparenthesis))$

- *Let $\Gamma, \Delta \vdash t \otimes r : n + m$ as a consequence of $\Gamma \vdash t : n$, $\Delta \vdash r : m$ and rule $\otimes$. By the induction hypothesis, $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \llparenthesis n\rrparenthesis$, that is $\llparenthesis\Gamma\rrparenthesis, \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis t\rrparenthesis) : \mathsf{qbit}^n$, and $\llparenthesis\Delta\rrparenthesis, \mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis r\rrparenthesis) : \llparenthesis m\rrparenthesis$, that is, $\llparenthesis\Delta\rrparenthesis, \mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis)) : \mathsf{qbit} \vdash term(\llparenthesis r\rrparenthesis) : \mathsf{qbit}^m$. By rule $\otimes_I$, we have $\llparenthesis\Gamma\rrparenthesis, \llparenthesis\Delta\rrparenthesis$, $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)) : \mathsf{qbit}, \mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis)) : \mathsf{qbit} \vdash \langle term(\llparenthesis t\rrparenthesis), term(\llparenthesis r\rrparenthesis)\langle : \mathsf{qbit}^{n+m}$.*

  Notice that $\llparenthesis\Gamma, \Delta\rrparenthesis = \llparenthesis\Gamma\rrparenthesis, \llparenthesis\Delta\rrparenthesis$, $\llparenthesis n + m\rrparenthesis = \mathsf{qbit}^{n+m}$, $term(\llparenthesis t \otimes r\rrparenthesis) = \langle term(\llparenthesis t\rrparenthesis), term(\llparenthesis r\rrparenthesis)\rangle$, and $\mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis,\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis tr\rrparenthesis)) = \mathsf{FV}_{\llparenthesis\Gamma\rrparenthesis}(term(\llparenthesis t\rrparenthesis)), \mathsf{FV}_{\llparenthesis\Delta\rrparenthesis}(term(\llparenthesis r\rrparenthesis))$.

- *Let* $\Gamma \vdash \pi^m t : (m,n)$ *as a consequence of* $\Gamma \vdash t : n$ *and rule* m. *By the induction hypothesis,* $(\!|\Gamma|\!), \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|t|\!))) : \mathsf{qbit} \vdash term((\!|t|\!)) : (\!|n|\!)$. *That is,* $(\!|\Gamma|\!), \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|\pi^m t|\!))) : \mathsf{qbit} \vdash term((\!|t|\!)) : \mathsf{qbit}^n$.

  It is easy to deduce by rules $\otimes_E$, $\otimes_I$, ax_1, *and* ax_2 *the following judgement:*

$$x_1 : \mathsf{qbit}, \dots, x_n : \mathsf{qbit} \vdash \mathsf{let}\ \langle b_1, \dots, b_m\rangle = \langle \mathsf{meas}\ x_1, \dots, \mathsf{meas}\ x_m\rangle \\ \mathsf{in}\ \langle b_1, \dots,\ b_m,\ \mathsf{new}\ b_1, \dots, \mathsf{new}\ b_m, x_{m+1}, \dots, x_n\rangle : \mathsf{bit}^m \otimes \mathsf{qbit}^n$$

  Hence, by rule $\otimes_E$,

$$(\!|\Gamma|\!), \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|t|\!))) : \mathsf{qbit} \vdash \mathsf{let}\ \langle x_1, \dots, x_n\rangle = term((\!|t|\!)) \\ \mathsf{in\ let}\ \langle b_1, \dots, b_m\rangle = \langle \mathsf{meas}\ x_1, \dots, \mathsf{meas}\ x_m\rangle \\ \mathsf{in}\ \langle b_1, \dots, b_m, \mathsf{new} b_1, \dots, \mathsf{new} b_m, x_{m+1}, \dots, x_n\rangle : \mathsf{bit}^m \otimes \mathsf{qbit}^n$$

  Notice that $(\!|n|\!) = \mathsf{qbit}^n$ *and* $\mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|\pi^m t|\!))) = \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|t|\!)))$.

- *Let* $\Delta \vdash (b^m, \rho^n) : (m,n)$ *as a consequence of rule* $\mathsf{ax_{am}}$ *and let* $term((\!|(b^m, \rho^n)|\!)) = \langle b_1, \dots, b_m, x_1, \dots, x_n\rangle$ *with* $b_j \in \{0,1\}$ *and* $x_i \in \mathcal{V}$. *By rule* ax_2, $\vdash b_i : \mathsf{bit}$ *for* $j = 1, \dots, m$. *By rule* ax_1, $x_i : \mathsf{qbit} \vdash x_i : \mathsf{qbit}$ *for* $i = 1, \dots, n$. *Then, by rule* $\otimes_I$, $\mathsf{FV}_{(\!|\Delta|\!)}(term((\!|(b^m, \rho^n)|\!))) : \mathsf{qbit} \vdash \langle b_1, \dots, b_m, x_1, \dots, x_n\rangle : \mathsf{bit}^m \otimes \mathsf{qbit}^n$. *Hence, by Lemma 2.1.2,* $(\!|\Delta|\!), \mathsf{FV}_{(\!|\Delta|\!)}(term((\!|(b^m, \rho^n)|\!))) : \mathsf{qbit} \vdash \langle b^m, \langle x_1, \dots, x_n\rangle\rangle : \mathsf{bit}^m \otimes \mathsf{qbit}^n$.

  Notice that $(\!|(m,n)|\!) = \mathsf{bit}^m \otimes \mathsf{qbit}^n$ *and* $\mathsf{FV}_{(\!|\Delta|\!)}(term((\!|(b^m, \rho^n)|\!))) = \{x_1, \dots, x_n\}$.

- *Let* $\Gamma, \Delta \vdash \mathsf{letcase}\ x = r\ \mathsf{in}\ t_1, \dots, t_{2^m} : A$ *as a consequence of* $\Delta, x : n \vdash t_1 : A, \dots, \Delta, x : n \vdash t_{2^m} : A$, $\Gamma \vdash r : (m,n)$, *and rule* lc.

  By the induction hypothesis, $(\!|\Delta, x : n|\!), \mathsf{FV}_{(\!|\Delta, x:n|\!)}(term((\!|t_i|\!))) : \mathsf{qbit} \vdash term((\!|t_i|\!)) : (\!|A|\!)$ *for* $i = 1, \dots, 2^m$, *and* $(\!|\Gamma|\!), \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|r|\!))) : \mathsf{qbit} \vdash term((\!|r|\!)) : (\!|(m,n)|\!)$. *That is,* $(\!|\Delta|\!), x : \mathsf{qbit}^n, \mathsf{FV}_{(\!|\Delta|\!), x:\mathsf{qbit}^n}(term((\!|t_i|\!))) : \mathsf{qbit} \vdash term((\!|t_i|\!)) : (\!|A|\!)$ *for* $i = 1, \dots, 2^m$, *and* $(\!|\Gamma|\!), \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|r|\!))) : \mathsf{qbit} \vdash term((\!|r|\!)) : \mathsf{bit}^m \otimes \mathsf{qbit}^n$.

  Let $V = \bigcup_{i=1}^{2^m} \mathsf{FV}_{(\!|\Delta|\!), x:\mathsf{qbit}^n}(term((\!|t_i|\!)))$. *By Lemma 2.1.2,* $(\!|\Delta|\!), V : \mathsf{qbit}, x : \mathsf{qbit}^n \vdash term((\!|t_i|\!)) : (\!|A|\!)$ *for* $i = 1, \dots, 2^m$.

  By rule ax_1, $b^m : \mathsf{bit}^m \vdash b^m : \mathsf{bit}^m$. *Then, by rule* case, $b : \mathsf{bit}^m, (\!|\Delta|\!), V : \mathsf{qbit}, x : \mathsf{qbit}^n \vdash \mathsf{case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)), \dots, term((\!|t_{2^m}|\!))\} : (\!|A|\!)$. *Then, by rule* $\otimes_E$ *we have*

$$(\!|\Gamma|\!), (\!|\Delta|\!), \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|r|\!))) : \mathsf{qbit}, V : \mathsf{qbit} \vdash \\ \mathsf{let}\ \langle b, x\rangle = term((\!|r|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)), \dots, term((\!|t_{2^n}|\!))\} : (\!|A|\!)$$

  Notice that $\mathsf{let}\ \langle b, x\rangle = term((\!|r|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)), \dots, term((\!|t_{2^n}|\!))\} = term((\!|\mathsf{letcase}\ x = r\ \mathsf{in}\ t_1, \dots, t_{2^m}|\!))$, $(\!|\Delta, x : n|\!) = (\!|\Delta|\!), x : \mathsf{qbit}^n$, $(\!|\Gamma, \Delta|\!) = (\!|\Gamma|\!), (\!|\Delta|\!)$, $\mathsf{FV}((\!|r|\!)) \cap (\bigcup_{i=1}^{2^m} \mathsf{FV}(t_i)) = \emptyset$, *and* $\mathsf{FV}_{(\!|\Delta|\!), x:\mathsf{qbit}^n}(term((\!|t_i|\!))) \cup \mathsf{FV}_{(\!|\Gamma|\!)}(term((\!|r|\!))) = \mathsf{FV}_{(\!|\Gamma|\!),(\!|\Delta|\!)}(term((\!|\mathsf{letcase}\ x = r\ \mathsf{in}\ t_1, \dots, t_{2^m}|\!)))$. $\square$

We now prove that the translation preserves the operational semantics. If a term in λ_ρ reduces to another with some probability, then the translation of the former reduces with the same probability to the translation of the latter. We take advantage of this theorem in Corollary 3.1.10 to prove the strong normalization property for λ_ρ.

We first prove the following lemma, stating that the translation of a value from λ_ρ is a value in λ_q.

Lemma 3.1.7 (Value preservation) *If $v \in V_\rho$, then $term(\llparenthesis v \rrparenthesis) \in V_q$.*

Proof *We proceed by induction on the definition of V_ρ.*

- *Let $v = x$. Notice that $term(\llparenthesis x \rrparenthesis) = x \in V_q$.*
- *Let $v = \lambda x.t$. Notice that $term(\llparenthesis \lambda x.t \rrparenthesis) = \lambda x.term(\llparenthesis t \rrparenthesis) \in V_q$.*
- *Let $v = \rho^n : n$. Notice that $term(\llparenthesis \rho^n \rrparenthesis) = \langle x_1, \dots, x_n \rangle \in V_q$.*
- *Let $v = (b^m, \rho^n)$. Notice that $\llparenthesis (b^m, \rho^n) \rrparenthesis = \langle b, x_1, \dots, x_n \rangle \in vSV$.* □

Then we need to prove that the translation commutes with substitution. That is, translating a term after performing a substitution is equivalent to translating the original term and the substituting.

Lemma 3.1.8 (Substitution) *Given $t, r \in \Lambda_\rho$, $\llparenthesis t[r/x] \rrparenthesis = \llparenthesis t \rrparenthesis [\llparenthesis r \rrparenthesis / x]$*

Proof *We proceed by structural induction on t.*

- *Let $t = y$.*
  - *If $x = y$, notice that $\llparenthesis x \rrparenthesis = [*,\ \varnothing,\ x]$ and $\llparenthesis (x[r/x]) \rrparenthesis = \llparenthesis r \rrparenthesis = \llparenthesis x \rrparenthesis [\llparenthesis r \rrparenthesis / x]$*
  - *If $y \neq x$, notice that $\llparenthesis y[r/x] \rrparenthesis = \llparenthesis y \rrparenthesis = \llparenthesis y \rrparenthesis [\llparenthesis r \rrparenthesis / x]$.*
- *Let $t = \lambda y.t'$. We assume, without loss of generality, that $x \neq y$. By the induction hypothesis, $\llparenthesis t'[r/x] \rrparenthesis = \llparenthesis t' \rrparenthesis [\llparenthesis r \rrparenthesis / x]$*

  Therefore,

$$\begin{aligned}
\llparenthesis (\lambda y.t')[r/x] \rrparenthesis &= \llparenthesis \lambda y.t'[r/x] \rrparenthesis \\
&= [st(\llparenthesis t'[r/x] \rrparenthesis), lnk(\llparenthesis t'[r/x] \rrparenthesis), \lambda y.\ term(\llparenthesis t'[r/x] \rrparenthesis)] \\
&= [st(\llparenthesis t' \rrparenthesis) \otimes st(\llparenthesis r \rrparenthesis), lnk(\llparenthesis t' \rrparenthesis) \cup lnk(\llparenthesis r \rrparenthesis), \\
&\quad \lambda y.term(\llparenthesis t' \rrparenthesis)[term(\llparenthesis r \rrparenthesis)/x]] \\
&= \llparenthesis \lambda y.t' \rrparenthesis [\llparenthesis r \rrparenthesis / x]
\end{aligned}$$

- *Let $t = t_1\ t_2$.*
  - *If $x \in \mathsf{FV}(t_1)$, then $x \notin \mathsf{FV}(t_2)$ by the linearity property. By the induction hypothesis, $\llparenthesis t_1[r/x] \rrparenthesis = \llparenthesis t_1 \rrparenthesis [\llparenthesis r \rrparenthesis / x]$ Therefore,*

$$\begin{aligned}
\llparenthesis (t_1\ t_2)[r/x] \rrparenthesis &= \llparenthesis t_1[r/x]\ t_2 \rrparenthesis \\
&= [st(\llparenthesis t_1[r/x] \rrparenthesis) \otimes st(\llparenthesis t_2 \rrparenthesis), lnk(\llparenthesis t_1[r/x] \rrparenthesis) \cup lnk(\llparenthesis t_2 \rrparenthesis), \\
&\quad term(\llparenthesis t_1[r/x] \rrparenthesis)\ term(\llparenthesis t_2 \rrparenthesis)] \\
&= [st(\llparenthesis t_1 \rrparenthesis) \otimes st(\llparenthesis r \rrparenthesis) \otimes st(\llparenthesis t_2 \rrparenthesis), lnk(\llparenthesis t_1 \rrparenthesis) \cup lnk(\llparenthesis r \rrparenthesis) \cup lnk(\llparenthesis t_2 \rrparenthesis), \\
&\quad term(\llparenthesis t_1 \rrparenthesis)[term(\llparenthesis r \rrparenthesis)/x]\ term(\llparenthesis t_2 \rrparenthesis)] \\
&= [st(\llparenthesis t_1 \rrparenthesis) \otimes st(\llparenthesis t_2 \rrparenthesis) \otimes st(\llparenthesis r \rrparenthesis), lnk(\llparenthesis t_1 \rrparenthesis) \cup lnk(\llparenthesis t_2 \rrparenthesis) \cup lnk(\llparenthesis r \rrparenthesis), \\
&\quad (term(\llparenthesis t_1 \rrparenthesis)\ term(\llparenthesis t_2 \rrparenthesis))[term(\llparenthesis r \rrparenthesis)/x]] \\
&= \llparenthesis t_1\ t_2 \rrparenthesis [\llparenthesis r \rrparenthesis / x]
\end{aligned}$$

– *If* $x \notin \mathsf{FV}(t_1)$, *by the induction hypothesis,* $(\!|t_2[r/x]|\!) = (\!|t_2|\!)[(\!|r|\!)/x]$ *Therefore,*

$$\begin{aligned}
(\!|(t_1\ t_2)[r/x]|\!) &= (\!|t_1\ t_2[r/x]|\!) \\
&= [st((\!|t_1|\!)) \otimes st((\!|t_2[r/x]|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2[r/x]|\!)), \\
&\quad\ term((\!|t_1|\!))\ term((\!|t_2[r/x]|\!))] \\
&= [st((\!|t_1|\!)) \otimes st((\!|t_2|\!)) \otimes st((\!|r|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)) \cup lnk((\!|r|\!)), \\
&\quad\ term((\!|t_1|\!))\ term((\!|t_2|\!))[term((\!|r|\!))/x]] \\
&= [st((\!|t_1|\!)) \otimes st((\!|t_2|\!)) \otimes st((\!|r|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)) \cup lnk((\!|r|\!)), \\
&\quad\ (term((\!|t_1|\!))\ term((\!|t_2|\!)))[term((\!|r|\!))/x]] \\
&= (\!|t_1\ t_2|\!)[(\!|r|\!)/x]
\end{aligned}$$

- *Let* $t = \rho^n$. *Notice that* x *is not a free variable in* $(\!|\rho^n|\!)$ *and therefore* $(\!|\rho^n[r/x]|\!) = (\!|\rho^n|\!) = (\!|\rho^n|\!)[(\!|r|\!)/x]$.

- *Let* $t = U^n\ t'$. *By the induction hypothesis,* $(\!|t'[r/x]|\!) = (\!|t'|\!)[(\!|r|\!)/x]$ *Therefore,*

$$\begin{aligned}
(\!|(U^n\ t')[r/x]|\!) &= (\!|U^n\ t'[r/x]|\!) \\
&= [st((\!|t'[r/x]|\!)), lnk((\!|t'[r/x]|\!)), U^n\ term((\!|t'[r/x]|\!))] \\
&= [st((\!|t'|\!)) \otimes st((\!|r|\!)), lnk((\!|t'|\!)) \cup lnk((\!|r|\!)), \\
&\quad\ U^n\ term((\!|t'|\!))[term((\!|r|\!))/x]] \\
&= (\!|U^n\ t'|\!)[(\!|r|\!)/x]
\end{aligned}$$

- *Let* $t = \pi^n\ t'$. *By the induction hypothesis,* $(\!|t'[r/x]|\!) = (\!|t'|\!)[(\!|r|\!)/x]$. *Therefore,*

$$\begin{aligned}
(\!|(\pi^n\ t')[r/x]|\!) &= (\!|\pi^n\ t'[r/x]|\!) \\
&= [st((\!|t'[r/x]|\!)), lnk((\!|t'[r/x]|\!)), \mathsf{let}\ \langle x_1, \ldots, x_n\rangle = term((\!|t'[r/x]|\!)) \\
&\qquad\quad \mathsf{in\ let}\ \langle b_1,\ \ldots,\ b_m\rangle = \langle meas\ x_1,\ \ldots,\ meas\ x_m\rangle \\
&\qquad\qquad \mathsf{in}\ \langle b_1,\ \ldots,\ b_n,\ new\ b_1,\ \ldots,\ new\ b_n,\ x_{m+1},\ \ldots,\ x_n\rangle] \\
&= [st((\!|t'|\!)) \otimes st((\!|r|\!)), lnk((\!|t'|\!)) \cup lnk((\!|r|\!)), \\
&\quad\ \mathsf{let}\ \langle x_1, \ldots, x_n\rangle = term((\!|t'|\!))[term((\!|r|\!))/x] \\
&\qquad\quad \mathsf{in\ let}\ \langle b_1,\ \ldots,\ b_m\rangle = \langle meas\ x_1,\ \ldots,\ meas\ x_m\rangle \\
&\qquad\qquad \mathsf{in}\ \langle b_1,\ \ldots,\ b_n,\ new\ b_1,\ \ldots,\ new\ b_n,\ x_{m+1},\ \ldots,\ x_n\rangle] \\
&= [st((\!|t'|\!)) \otimes st((\!|r|\!)), lnk((\!|t'|\!)) \cup lnk((\!|r|\!)), \\
&\quad\ (\mathsf{let}\ \langle x_1, \ldots, x_n\rangle = term((\!|t'|\!)) \\
&\qquad\quad \mathsf{in\ let}\ \langle b_1,\ \ldots,\ b_m\rangle = \langle meas\ x_1,\ \ldots,\ meas\ x_m\rangle \\
&\qquad\qquad \mathsf{in}\ \langle b_1,\ \ldots,\ b_n,\ new\ b_1,\ \ldots,\ new\ b_n,\ x_{m+1},\ \ldots,\ x_n\rangle) \\
&\quad\ [term((\!|r|\!))/x]] \\
&= (\!|\pi^n\ t'|\!)[(\!|r|\!)/x]
\end{aligned}$$

- *Let* $t = t_1 \otimes t_2$.

  – *If* $x \in \mathsf{FV}(t_1)$, *then* $x \notin \mathsf{FV}(t_2)$ *by the linearity property.*

By the induction hypothesis, $(\!|t_1[r/x]|\!) = (\!|t_1|\!)[(\!|r|\!)/x]$ *Therefore,*

$$\begin{aligned}(\!|(t_1 \otimes t_2)[r/x]|\!) &= (\!|t_1[r/x] \otimes t_2|\!)\\ &= [st((\!|t_1[r/x]|\!)) \otimes st((\!|t_2|\!)), lnk((\!|t_1[r/x]|\!)) \cup lnk((\!|t_2|\!)),\\ &\quad term((\!|t_1[r/x]|\!)) \otimes term((\!|t_2|\!))]\\ &= [st((\!|t_1|\!)) \otimes st((\!|r|\!)) \otimes st((\!|t_2|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|r|\!)) \cup lnk((\!|t_2|\!)),\\ &\quad term((\!|t_1|\!))[term((\!|r|\!))/x] \otimes term((\!|t_2|\!))]\\ &= [st((\!|t_1|\!)) \otimes st((\!|t_2|\!)) \otimes st((\!|r|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)) \cup lnk((\!|r|\!)),\\ &\quad (term((\!|t_1|\!)) \otimes term((\!|t_2|\!)))[term((\!|r|\!))/x]]\\ &= (\!|t_1 \otimes t_2|\!)[(\!|r|\!)/x]\end{aligned}$$

– *If* $x \notin \mathsf{FV}(t_1)$*, by the induction hypothesis,* $(\!|t_2[r/x]|\!) = (\!|t_2|\!)[(\!|r|\!)/x]$
  Therefore,

$$\begin{aligned}(\!|(t_1 \otimes t_2)[r/x]|\!) &= (\!|t_1 \otimes t_2[r/x]|\!)\\ &= [st((\!|t_1|\!)) \otimes st((\!|t_2[r/x]|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2[r/x]|\!)),\\ &\quad term((\!|t_1|\!)) \otimes term((\!|t_2[r/x]|\!))]\\ &= [st((\!|t_1|\!)) \otimes st((\!|t_2|\!)) \otimes st((\!|r|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)) \cup lnk((\!|r|\!)),\\ &\quad term((\!|t_1|\!)) \otimes term((\!|t_2|\!))[term((\!|r|\!))/x]]\\ &= [st((\!|t_1|\!)) \otimes st((\!|t_2|\!)) \otimes st((\!|r|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)) \cup lnk((\!|r|\!)),\\ &\quad (term((\!|t_1|\!)) \otimes term((\!|t_2|\!)))[term((\!|r|\!))/x]]\\ &= (\!|t_1 \otimes t_2|\!)[(\!|r|\!)/x]\end{aligned}$$

- *Let* $t = (b, t')$*. By the induction hypothesis,* $(\!|t'[r/x]|\!) = (\!|t'|\!)[(\!|r|\!)/x]$ *Therefore,*

$$\begin{aligned}(\!|(b,\ t')[r/x]|\!) &= (\!|(b,\ t'[r/x])|\!)\\ &= [st((\!|t'[r/x]|\!)), lnk((\!|t'[r/x]|\!)), \langle b, term((\!|t'[r/x]|\!))\rangle]\\ &= [st((\!|t'|\!)) \otimes st((\!|r|\!)), lnk((\!|t'|\!)) \cup lnk((\!|r|\!)),\\ &\quad \langle b, term((\!|t'|\!))[term((\!|r|\!))/x]\rangle]\\ &= (\!|(b,\ t')|\!)[(\!|r|\!)/x]\end{aligned}$$

- *Let* $t = \mathsf{letcase}\ y = s\ \mathsf{in}\ \{t_1, \ldots, t_{2^n}\}$*. We assume, without loss of generality, that* $x \neq y$*.*

  – *If there is an* $i \in \{1, \ldots, 2^n\}$ *such that* $x \in \mathsf{FV}(t_i)$*, then by the linearity property* $x \notin \mathsf{FV}(t_j)\ \forall j \neq i$ *and* $x \notin \mathsf{FV}(s)$*.*

    By the induction hypothesis, $(\!|t_i[r/x]|\!) = (\!|t_i|\!)[(\!|r|\!)/x]$*.*

Therefore,

$$
\begin{aligned}
&(\!|(\mathsf{letcase}\ y = s\ \mathsf{in}\ \{t_1,\ldots,t_{2^n}\})[r/x]|\!) \\
&\quad = (\!|(\mathsf{letcase}\ y = s\ \mathsf{in}\ \{t_1,\ldots,t_i[r/x],\ldots,t_{2^n}\})|\!) \\
&\quad = [st((\!|s|\!)) \otimes \bigotimes_{\substack{j=1\\ j\neq i}}^{2^n} st((\!|t_j|\!)) \otimes st((\!|t_i[r/x]|\!)), \\
&\qquad lnk((\!|s|\!)) \cup \bigcup_{\substack{j=1\\ j\neq i}}^{2^n} lnk((\!|t_j|\!)) \cup lnk((\!|t_i[r/x]|\!)), \\
&\qquad \mathsf{let}\ \langle b,y\rangle = term((\!|s|\!))\ \mathsf{in\ case}\ b\ \mathsf{of} \\
&\qquad\qquad \{term((\!|t_1|\!)),\ldots,term((\!|t_i[r/x]|\!)),\ldots,term((\!|t_{2^n}|\!))\}] \\
&\quad = [st((\!|s|\!)) \otimes \bigotimes_{j=1}^{2^n} st((\!|t_j|\!)) \otimes st((\!|r|\!)), \\
&\qquad lnk((\!|s|\!)) \cup \bigcup_{j=1}^{2^n} lnk((\!|t_j|\!)) \cup lnk((\!|r|\!)), \\
&\qquad \mathsf{let}\ \langle b,y\rangle = term((\!|s|\!))\ \mathsf{in\ case}\ b\ \mathsf{of} \\
&\qquad\qquad \{term((\!|t_1|\!)),\ldots,term((\!|t_i|\!))[term((\!|r|\!))/x],\ldots,term((\!|t_{2^n}|\!))\}] \\
&\quad = (\!|\mathsf{letcase}\ y = s\ \mathsf{in}\ \{t_1,\ldots,t_{2^n}\}|\!)[(\!|r|\!)/x]
\end{aligned}
$$

– *If* $\forall i,\ x \notin \mathsf{FV}(t_i)$. *By the induction hypothesis,* $(\!|s[r/x]|\!) = (\!|s|\!)[(\!|r|\!)/x]$. *Hence,*

$$
\begin{aligned}
&(\!|(\mathsf{letcase}\ y = s\ \mathsf{in}\ \{t_1,\ldots,t_{2^n}\})[r/x]|\!) \\
&\quad = (\!|(\mathsf{letcase}\ y = s[r/x]\ \mathsf{in}\ \{t_1,\ldots,t_{2^n}\})|\!) \\
&\quad = [st((\!|s[r/x]|\!)) \otimes \bigotimes_{i=1}^{2^n} st((\!|t_i|\!)), lnk((\!|s[r/x]|\!)) \cup \bigcup_{i=1}^{2^n} lnk((\!|t_i|\!)), \\
&\qquad \mathsf{let}\ \langle b,y\rangle = term((\!|s[r/x]|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)),\ldots,term((\!|t_{2^n}|\!))\}] \\
&\quad = [st((\!|s|\!)) \otimes \bigotimes_{i=1}^{2^n} st((\!|t_i|\!)) \otimes st((\!|r|\!)), \\
&\qquad lnk((\!|s|\!)) \cup \bigcup_{i=1}^{2^n} lnk((\!|t_i|\!)) \cup lnk((\!|r|\!)), \\
&\qquad \mathsf{let}\ \langle b,y\rangle = term((\!|s|\!))[term((\!|r|\!))/x]\ \mathsf{in} \\
&\qquad\qquad \mathsf{case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)),\ldots,term((\!|t_{2^n}|\!))\}] \\
&\quad = (\!|\mathsf{letcase}\ y = s\ \mathsf{in}\ \{t_1,\ldots,t_{2^n}\}|\!)[(\!|r|\!)/x] \qquad \square
\end{aligned}
$$

We can now prove the following theorem:

Theorem 3.1.9 *Let* $t, r \in \Lambda_\rho$. *If* $t \longrightarrow_p r$, *then* $(\!|t|\!) \hookrightarrow_p^+ (\!|r|\!)$.

Proof *We proceed by induction on the relation* $\longrightarrow_p$.

- *Let* $t = (\lambda x.\ s)v$, $r = s[v/x]$, *and* $p = 1$ *with* $v \in V_\rho$. *Notice that*
  $(\!|t|\!) = [st((\!|s|\!)) \otimes st((\!|v|\!)), lnk((\!|s|\!)) \cup lnk((\!|v|\!)), (\lambda x.\ term((\!|s|\!)))term((\!|v|\!))]$
  $(\!|r|\!) = [st((\!|s[v/x]|\!)), lnk((\!|s[v/x]|\!)), term((\!|s[v/x]|\!))]$

  By Lemma 3.1.7, $term((\!|v|\!))$ *is a value. Therefore,* $(\!|t|\!) \hookrightarrow_1 [st((\!|s|\!)) \otimes st((\!|v|\!)), lnk((\!|s|\!)) \cup lnk((\!|v|\!)), term((\!|s|\!))[term((\!|v|\!)/x)]]$.

  By Lemma 3.1.8, $term((\!|s|\!))[term((\!|v|\!)/x)] = term((\!|s[v/x]|\!)) = term((\!|r|\!))$.

- *Let* $t = U^m\ \rho^n$, $r = \rho'^n$, *and* $p = 1$ *where* $\rho'^n = \overline{U^m}\rho^n\overline{U^m}^\dagger$, $\mathsf{pur}(\rho^n) = |\phi\rangle\langle\phi|$ *and* $\mathsf{pur}(\rho'^n) = |\phi'\rangle\langle\phi'|$. *Notice that* $(\!|t|\!) = [|\phi\rangle, \{x_i \mapsto i\}_{i=1}^n, U\langle x_1, \ldots, x_n\rangle] \hookrightarrow_1 [|\phi'\rangle, \{x_i \mapsto i\}_{i=1}^n, \langle x_1, \ldots, x_n\rangle] = (\!|r|\!)$.

- *Let* $t = \pi^m\rho^n$ *and* $r = (i, \rho_i^n)$, *where* $\mathsf{pur}(\rho) = |\phi\rangle\langle\phi|$, $\mathsf{pur}(\rho_i^n) = |\phi_i\rangle\langle\phi_i|$, *and* $i^1, \ldots, i^m$ *is the binary encoding of* i. *Notice that*

$$
\begin{aligned}
(\!|t|\!) = [|\phi\rangle, \{x_i \mapsto i\}_{i=1}^n, &\ \mathsf{let}\ \langle y_1, \ldots, y_n\rangle = \langle x_1, \ldots, x_n\rangle \\
&\ \mathsf{in\ let}\ \langle b_1, \ldots, b_m\rangle = \langle meas\ y_1, \ldots, meas\ y_m\rangle \\
&\ \quad \mathsf{in}\ \langle b_1, \ldots, b_m, new\ b_1, \ldots, new\ b_m, y_{m+1}, \ldots, y_n\rangle] \\
\hookrightarrow_1 [|\phi\rangle, \{x_i \mapsto i\}_{i=1}^n, &\ \mathsf{let}\ \langle b_1, \ldots, b_m\rangle = \langle meas\ x_1, \ldots, meas\ x_m\rangle \\
&\ \quad \mathsf{in}\ \langle b_1, \ldots, b_m,\ new b_1, \ldots, new\ b_m, x_{m+1}, \ldots, x_n\rangle] \\
\twoheadrightarrow_p [|\phi_i\rangle, \{x_i \mapsto i\}_{i=1}^n, &\ \mathsf{let}\ \langle b_1, \ldots, b_m\rangle = \langle i^1, \ldots, i^m\rangle \\
&\ \quad \mathsf{in}\ \langle b_1, \ldots, b_m, new\ b_1, \ldots, new\ b_m, x_{m+1}, \ldots, x_n\rangle] \\
\hookrightarrow_1 [|\phi_i\rangle, \{x_i \mapsto i\}_{i=1}^n, &\ \langle i^1, \ldots, i^m, new\ i^1, \ldots, new\ i^m, x_{m+1}, \ldots, x_n\rangle] \\
\twoheadrightarrow_1 [|\phi_i\rangle \otimes |i\rangle, &\ \{x_i \mapsto i\}_{i=1}^n \cup \{y_i \mapsto i\}_{i=1}^m, \langle i^1, \ldots, i^m, y_1, \ldots, y_m, x_{m+1}, \ldots, x_n\rangle] \\
= (\!|r|\!) &
\end{aligned}
$$

- *Let* $t = (\mathsf{letcase}\ x = (b^m, \rho^n)\ \mathsf{in}\ \{t_0, \ldots, t_{2^n-1}\})$, $r = t_{b^m}[\rho^n/x]$, *and* $p = 1$ *where* $\mathsf{pur}(\rho) = |\phi\rangle\langle\phi|$. *Notice that*

$$
\begin{aligned}
(\!|t|\!) = [|\phi\rangle \otimes \bigotimes_{i=1}^{2^n} st((\!|t_i|\!)), \{x_i \mapsto i\}_{i=1}^n \cup \bigcup_{i=1}^{2^n} lnk((\!|t_i|\!)), \\
\mathsf{let}\ \langle b, x\rangle = \langle b^m,\ x_1,\ \ldots,\ x_n\rangle\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)), \ldots, term((\!|t_{2^n}|\!))\}] \\
\twoheadrightarrow_1 [|\phi\rangle \otimes \bigotimes_{i=1}^{2^n} st((\!|t_i|\!)), \{x_i \mapsto i\}_{i=1}^n \cup \bigcup_{i=1}^{2^n} lnk((\!|t_i|\!)), term((\!|t_{b_m}|\!))[\langle x_1,\ \ldots,\ x_n\rangle/x]] \\
= (\!|r|\!)
\end{aligned}
$$

- *Let* $t = s\,t'$ *and* $r = s\,r'$, *where* $t' \longrightarrow_p r'$. *By the induction hypothesis,* $(\!|t'|\!) \hookrightarrow_p^+ (\!|r'|\!)$. *Therefore*

$$
\begin{aligned}
(\!|t|\!) &= [st((\!|s|\!)) \otimes st((\!|t'|\!)), lnk((\!|s|\!)) \cup lnk((\!|t'|\!)), term(term((\!|s|\!))\ (\!|t'|\!))] \\
&\hookrightarrow_p^+ [st((\!|s|\!)) \otimes st((\!|r'|\!)), lnk((\!|s|\!)) \cup lnk((\!|r'|\!)), term(term((\!|s|\!))\ (\!|r'|\!))] \\
&= (\!|r|\!)
\end{aligned}
$$

- *Let $t = t'\, s$ and $r = r'\, s$ where $s \in V_\rho$ and $t' \longrightarrow_p r'$. By Lemma 3.1.7, $(\!|s|\!)$ is a value. By the induction hypothesis, $(\!|t'|\!) \hookrightarrow_p^+ (\!|r'|\!)$. Therefore*

$$\begin{aligned}(\!|t|\!) &= [st((\!|t'|\!)) \otimes st((\!|s|\!)), lnk((\!|t'|\!)) \cup lnk((\!|s|\!)), term(term((\!|t'|\!))\, (\!|s|\!))] \\ &\hookrightarrow_p^+ [st((\!|r'|\!)) \otimes st((\!|s|\!)), lnk((\!|r'|\!)) \cup lnk((\!|s|\!)), term(term((\!|r'|\!))\, (\!|s|\!))] \\ &= (\!|r|\!)\end{aligned}$$

- *Let $t = U^n\, t'$ and $r = U^n\, r'$, where $t' \longrightarrow_p r'$. By the induction hypothesis, $(\!|t'|\!) \hookrightarrow_p^+ (\!|r'|\!)$. Therefore $(\!|t|\!) = [st((\!|t'|\!)), lnk((\!|t'|\!)), U^n\, term((\!|t'|\!))] \hookrightarrow_p^+ [st((\!|r'|\!)), lnk((\!|r'|\!)), U^n\, term((\!|r'|\!))] = (\!|r|\!)$.*

- *Let $t = \pi^n\, t'$ and $r = \pi^n\, r'$, where $t' \longrightarrow_p r'$. By the induction hypothesis, $(\!|t'|\!) \hookrightarrow_p^+ (\!|r'|\!)$. Therefore*

$$\begin{aligned}(\!|t|\!) = [&st((\!|t'|\!)), lnk((\!|t'|\!)), \mathsf{let}\ \langle x_1, \ldots, x_n\rangle = term((\!|t'|\!)) \\ &\quad \mathsf{in\ let}\ \langle b_1, \ldots, b_m\rangle = \langle meas\ x_1, \ldots, meas\ x_m\rangle \\ &\quad\quad \mathsf{in}\ \langle b_1, \ldots, b_m, new\ b_1, \ldots, new\ b_m, x_{m+1}, \ldots, x_n\rangle] \\ \hookrightarrow_p^+ [&st((\!|r'|\!)), lnk((\!|r'|\!)), \mathsf{let}\ \langle x_1, \ldots, x_n\rangle = term((\!|r'|\!)) \\ &\quad \mathsf{in\ let}\ \langle b_1, \ldots, b_m\rangle = \langle meas\ x_1, \ldots, meas\ x_m\rangle \\ &\quad\quad \mathsf{in}\ \langle b_1, \ldots, b_m, new\ b_1, \ldots, new\ b_m, x_{m+1}, \ldots, x_n\rangle] \\ = (\!|r|\!)&\end{aligned}$$

- *Let $t = (\!|\mathsf{letcase}\ x = t'\ \mathsf{in}\ \{s_1, \ldots, s_{2^n}\}|\!)$ and $r = (\!|\mathsf{letcase}\ x = r'\ \mathsf{in}\ \{s_1, \ldots, s_{2^n}\}|\!)$, where $t' \to_p r'$. By the induction hypothesis, $(\!|t'|\!) \hookrightarrow_p^+ (\!|r'|\!)$. Therefore*

$$\begin{aligned}(\!|t|\!) &= [st((\!|t'|\!)) \otimes \bigotimes_{i=1}^{2^n} st((\!|s_i|\!)), lnk((\!|t'|\!)) \cup \bigcup_{i=1}^{2^n} lnk((\!|s_i|\!)), \\ &\qquad \mathsf{let}\ \langle b, x\rangle = term((\!|t'|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|s_1|\!)), \ldots, term((\!|s_{2^n}|\!))\}] \\ &\hookrightarrow_p^+ [st((\!|r'|\!)) \otimes \bigotimes_{i=1}^{2^n} st((\!|s_i|\!)), lnk((\!|r'|\!)) \cup \bigcup_{i=1}^{2^n} lnk((\!|s_i|\!)), \\ &\qquad \mathsf{let}\ \langle b, x\rangle = term((\!|r'|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|s_1|\!)), \ldots, term((\!|s_{2^n}|\!))\}] \\ &= (\!|r|\!)\end{aligned}$$ □

As a corollary of Theorem 3.1.9, we can prove the strong normalization of λ_ρ. This property states that any well-typed term cannot be reduced indefinitely. That is, by reducing a term repeatedly we eventually reach a value.

Corollary 3.1.10 (Strong normalization) *λ_ρ is strong normalizing.*

Proof *By contradiction. Let $t_1 \in \Lambda_\rho$ be a well-typed term and let $t_1 \to_{p_1} t_2 \to_{p_2} \ldots$ be an infinite reduction.*

By Theorem 3.1.9, $(\!|t_i|\!) \to_{p_i}^+ (\!|t_{i+1}|\!)$ for all i.

Therefore there exists an infinite reduction in λ_q, but, by [SV08, Theorem 3.8], λ_q is strong normalizing, which constitutes a contradiction.

Therefore, λ_ρ is strong normalizing. □

3.2 Retraction from λ_q to λ_ρ

Having the quantum state represented with density matrices directly within the terms in λ_ρ allows us to define the generalization of terms used in λ_ρ^o easily, but it does not let us separate entangled qubits and operate on them in separate parts of a term, as we can do in λ_q.

For example, the following term in λ_q has no direct counterpart in λ_ρ:

$$r = [\beta_{00},\ \{x_i \mapsto i\}_{i=1}^{2},\ (t_1\, x_1)\,(t_2\, x_2)]$$

A translation of this quantum closure to λ_ρ would require encoding the entangled quantum state as a single density matrix, but we cannot join the qubit pointers x_1 and x_2 directly because the term uses them as arguments in different applications.

Therefore, we cannot define a general inverse translation from λ_q to λ_ρ. We can, however, introduce a left inverse (also called a *retraction*) for the translation defined in Section 3.1, to prove that $(\!|\cdot|\!)$ does not lose any information from the original terms.

$$(\!|\cdot|\!)^{-1}\ :\ Im((\!|\cdot|\!)) \to \Lambda_\rho$$

We inductively define the left-inverse in Table 3.3.

$$
\begin{aligned}
(\!|[\left|\phi\right\rangle, L, x]|\!)^{-1} &= x \\
(\!|[\left|\phi\right\rangle, L, \lambda x.\ t]|\!)^{-1} &= \lambda x.\ (\!|[\left|\phi\right\rangle, L, t]|\!)^{-1} \\
(\!|[\left|\phi\right\rangle, L, t_1\ t_2]|\!)^{-1} &= (\!|[\left|\phi\right\rangle, L, t_1]|\!)^{-1}\ (\!|[\left|\phi\right\rangle, L, t_2]|\!)^{-1} \\
(\!|[\left|\phi\right\rangle, L, \langle x_1, \ldots, x_n\rangle]|\!)^{-1} &= tr_{E_n}(\left|\phi\right\rangle\!\left\langle\phi\right|) \\
&\quad \text{where } x_i\ :\mathsf{qbit} \text{ and } tr_{E_n} \text{ is the trace over the first } n \text{ qubits} \\
(\!|[\left|\phi\right\rangle, L, U\ t]|\!)^{-1} &= U\ (\!|[\left|\phi\right\rangle, L, t]|\!)^{-1} \\
(\!|[\left|\phi\right\rangle, L, \langle t_1, t_2\rangle]|\!)^{-1} &= (\!|[\left|\phi\right\rangle, L, t_1]|\!)^{-1} \otimes (\!|[\left|\phi\right\rangle, L, t_2]|\!)^{-1} \qquad \text{where } t_1 : \mathsf{qbit}^n \text{ and } t_2 : \mathsf{qbit}^m \\
(\!|[\left|\phi\right\rangle, L, \mathsf{let}\ \langle x_1,\ \ldots,\ x_n\rangle &= t\ \mathsf{in} \\
\mathsf{let}\ \langle b_1,\ \ldots,\ b_m\rangle &= \langle meas\ x_1,\ \ldots,\ meas\ x_m\rangle\ \mathsf{in} \\
\langle b_1,\ \ldots,\ b_m,\ new\ b_1,\ \ldots,\ new\ b_m,\ &x_{m+1},\ \ldots,\ x_n\rangle]|\!)^{-1} \\
&= \pi^m\ (\!|[\left|\phi\right\rangle, L, t]|\!)^{-1} \\
(\!|[\left|\phi\right\rangle, L, \langle b_1, \ldots, b_m, x_1, \ldots, x_n\rangle]|\!)^{-1} &= (b^m, (\!|[\left|\phi\right\rangle, L, \langle x_1, \ldots, x_n\rangle]|\!)^{-1}) \qquad \text{where } b_i : \mathsf{bit} \text{ and } x_j : \mathsf{qbit} \\
(\!|[\left|\phi\right\rangle, L, \mathsf{let}\ \langle b, x\rangle = t\ \mathsf{in}\ \mathsf{case}\ b\ \mathsf{of}\ &\{t_1, \ldots, t_{2^n}\}]|\!)^{-1} \\
&= (\mathsf{letcase}\ x = (\!|[\left|\phi\right\rangle, L, t]|\!)^{-1}\ \mathsf{in} \\
&\qquad \{(\!|[\left|\phi\right\rangle, L, t_1]|\!)^{-1}, \ldots, (\!|[\left|\phi\right\rangle, L, t_{2^n}]|\!)^{-1}\})
\end{aligned}
$$

Table 3.3: Left-inverse of $(\!|\cdot|\!)$

Below we prove that this definition is effectively a left inverse. That is, that $(\!|\cdot|\!)$ and $(\!|\cdot|\!)^{-1}$ compose to the identity.

Lemma 3.2.1 *Let $t \in \Lambda_\rho$, then $t = (\!|(\!|t|\!)|\!)^{-1}$*

Proof *We proceed by induction on t.*

- *Let* $t = x$. *Notice that* $(\!|x|\!) = [*,\, \varnothing,\, x]$ *and* $(\!|[*,\, \varnothing,\, x]|\!)^{-1} = x$
- *Let* $t = \lambda x.t'$. *By the induction hypothesis,* $(\!|(\!|t'|\!)|\!)^{-1} = t'$.
  Notice that $(\!|(\!|\lambda x.t'|\!)|\!)^{-1} = (\!|[st((\!|t'|\!)), lnk((\!|t'|\!)), \lambda x.\ term((\!|t'|\!))]|\!)^{-1} = \lambda x.(\!|(\!|t'|\!)|\!)^{-1} = \lambda x.t'$.
- *Let* $t = t_1\ t_2$. *By the induction hypothesis,* $(\!|(\!|t_1|\!)|\!)^{-1} = t_1$ *and* $(\!|(\!|t_2|\!)|\!)^{-1} = t_2$.
  Notice that $(\!|(\!|t_1\, t_2|\!)|\!)^{-1} = (\!|[st((\!|t_1|\!)) \otimes st((\!|t_2|\!)),\ lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)),\ term((\!|t_1|\!))\ term((\!|t_2|\!))]|\!)^{-1} = (\!|(\!|t_1|\!)|\!)^{-1}\ (\!|(\!|t_2|\!)|\!)^{-1} = t_1\, t_2$.
- *Let* $t = \rho^n$. *Notice that* $(\!|(\!|\rho^n|\!)|\!)^{-1} = (\!|[\mathsf{pur}(\rho^n),\ \{x_i \mapsto i\}_{i=1}^n,\ \langle x_1, \ldots, x_n\rangle]|\!)^{-1} = tr_{E_n}(\mathsf{pur}(\rho^n)) = \rho^n$.
- *Let* $t = U^n\, t'$. *By the induction hypothesis,* $(\!|(\!|t'|\!)|\!)^{-1} = t'$.
  Notice that $(\!|(\!|U^n\, t'|\!)|\!)^{-1} = (\!|[st((\!|t'|\!)), lnk((\!|t'|\!)), U^n\ term((\!|t'|\!))]|\!)^{-1} = U^n\ (\!|(\!|t'|\!)|\!)^{-1} = U^n\, t'$.
- *Let* $t = \pi^n\ t'$. *By the induction hypothesis,* $(\!|(\!|t'|\!)|\!)^{-1} = t'$.
  Notice that,

$$\begin{aligned}(\!|(\!|\pi^n\, t'|\!)|\!)^{-1} &= (\!|[st((\!|t'|\!)), lnk((\!|t'|\!)), \mathsf{let}\ \langle x_1, \ldots, x_n\rangle = term((\!|t'|\!))\ \mathsf{in}\\ &\qquad \mathsf{let}\ \langle b_1,\ \ldots,\ b_m\rangle = \langle meas\ x_1,\ \ldots,\ meas\ x_m\rangle\\ &\qquad\quad \mathsf{in}\ \langle b_1,\ \ldots,\ b_n,\ new\ b_1,\ \ldots,\ new\ b_n,\ x_{m+1},\ \ldots,\ x_n\rangle]|\!)^{-1}\\ &= \pi^n\ (\!|(\!|t'|\!)|\!)^{-1} = \pi^n\, t'\end{aligned}$$

  .
- *Let* $t = t_1 \otimes t_2$. *By the induction hypothesis,* $(\!|(\!|t_1|\!)|\!)^{-1} = t_1$ *and* $(\!|(\!|t_2|\!)|\!)^{-1} = t_2$.
  Notice that $(\!|(\!|t_1 \otimes t_2|\!)|\!)^{-1} = (\!|[st((\!|t_1|\!)) \otimes st((\!|t_2|\!)), lnk((\!|t_1|\!)) \cup lnk((\!|t_2|\!)), \langle term((\!|t_1|\!)), term((\!|t_2|\!))\rangle]|\!)^{-1} = (\!|(\!|t_1|\!)|\!)^{-1} \otimes (\!|(\!|t_2|\!)|\!)^{-1} = t_1 \otimes t_2$.
- *Let* $t = (b,\, t')$. *By the induction hypothesis,* $(\!|(\!|t'|\!)|\!)^{-1} = t'$.
  Notice that $(\!|(\!|(b,\, t')|\!)|\!)^{-1} = (\!|[st((\!|t'|\!)), lnk((\!|t'|\!)), (b,\ term((\!|t'|\!)))]|\!)^{-1} = (b,\ (\!|(\!|t'|\!)|\!)^{-1}) = (b,\, t')$.
- *Let* $t = \mathsf{letcase}\ x = r\ \mathsf{in}\ \{t_1, \ldots, t_{2^n}\}$. *By the induction hypothesis,* $(\!|(\!|r|\!)|\!)^{-1} = r$ *and* $(\!|(\!|t_i|\!)|\!)^{-1} = t_i$ *for* $1 \le i \le 2^n$.
  Notice that,

$$\begin{aligned}&(\!|(\!|\mathsf{letcase}\ x = r\ \mathsf{in}\ \{t_1, \ldots, t_{2^n}\}|\!)|\!)^{-1}\\ &\quad= (\!|[st((\!|r|\!)) \otimes \bigotimes_{i=1}^{2^n} st((\!|t_i|\!)), lnk((\!|r|\!)) \cup \bigcup_{i=1}^{2^n} lnk((\!|t_i|\!)),\\ &\qquad \mathsf{let}\ \langle b, y\rangle = term((\!|r|\!))\ \mathsf{in\ case}\ b\ \mathsf{of}\ \{term((\!|t_1|\!)), \ldots, term((\!|t_{2^n}|\!))\}]|\!)^{-1}\\ &\quad= \mathsf{letcase}\ x = (\!|(\!|r|\!)|\!)^{-1}\ \mathsf{in}\ \{(\!|(\!|t_1|\!)|\!)^{-1}, \ldots, (\!|(\!|t_{2^n}|\!)|\!)^{-1}\}\\ &\quad= \mathsf{letcase}\ x = r\ \mathsf{in}\ \{t_1, \ldots, t_{2^n}\}\end{aligned}$$

$\square$

3.3 Translation from λ_ρ^o to λ_q

We reuse the translation introduced in Section 3.1 to translate λ_ρ^o to λ_ρ by defining a translation from λ_ρ^o to λ_ρ, and then composing it with $\llparenthesis\cdot\rrparenthesis$.

We proceed to define the translation

$$\{\!\{\cdot\}\!\} \;:\; \Lambda_\rho^o \to \Lambda_\rho.$$

This translation is shallow in the constructors shared by both calculi. The superposition of terms is translated by explicitly measuring a new density matrix and choosing one of the translated terms using a letcase. We define it in Table 3.4.

$$\begin{aligned}
\{\!\{x\}\!\} &= x \\
\{\!\{\lambda x.\ t\}\!\} &= \lambda x.\ \{\!\{t\}\!\} \\
\{\!\{t_1\ t_2\}\!\} &= \{\!\{t_1\}\!\}\ \{\!\{t_2\}\!\} \\
\{\!\{\rho^n\}\!\} &= \rho^n \\
\{\!\{U^n\ t\}\!\} &= U^n\ \{\!\{t\}\!\} \\
\{\!\{\pi^m\ t\}\!\} &= \pi^m\ \{\!\{t\}\!\} \\
\{\!\{t_1 \otimes t_2\}\!\} &= \{\!\{t_1\}\!\} \otimes \{\!\{t_2\}\!\} \\
\{\!\{\mathsf{letcase}^o\ x = r\ \mathsf{in}\ \{t_1, \dots, t_{2^n}\}\}\!\} &= \mathsf{letcase}\ x = \{\!\{r\}\!\}\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \dots, \{\!\{t_{2^n}\}\!\}\} \\
\{\!\{\sum_{i=1}^{n} p_i t_i\}\!\} &= \mathsf{letcase}\ x = \pi^k\ \rho^k\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \dots, \{\!\{t_{2^k}\}\!\}\}
\end{aligned}$$

Where $k = \lceil \log_2(n) \rceil$, $t_{n+1} = \cdots = t_{2^k} = t_1$, and $\rho^k = \sum_{i=1}^{n} p_i \, |i\rangle\langle i|$.

Table 3.4: Translation from λ_ρ^o to λ_ρ

Example 3.3.1 *Let t be the coin flipping term in Example 2.3.1:*

$$t = \frac{1}{2} r_1 + \frac{1}{2} r_2.$$

This term can be translated as:

$$\{\!\{t\}\!\} = \mathsf{letcase}\ x = \pi^k \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}\ \mathsf{in}\ \{r_1, r_2\}$$

Since both calculi share the same set of types, we reuse the type translation.

Remark 3.3.2 *As shown in the following example (Example 3.3.3), this definition produces a non injective mapping. A* letcase$^{\mathsf{o}}$ *term may be translated to the same closure as a sum term.*

In fact, the image of translation corresponds to the intersection between the terms of λ_ρ and λ_ρ^o.

Example 3.3.3 (Translation collision) *Consider the terms*

$$t = \mathsf{letcase}^o\ x = \pi^1(\frac{1}{3}\,|0\rangle\langle 0| + \frac{2}{3}\,|1\rangle\langle 1|)\ \mathsf{in}\ \{\lambda xy.x, \lambda xy.y\}$$

and

$$r = \frac{1}{3}(\lambda xy.x) + \frac{2}{3}(\lambda xy.y)$$

Both terms translate to the same closure in λ_q:

$$\begin{aligned}\{\!\{r\}\!\} &= \mathsf{letcase}\ x = \pi^1(\frac{1}{3}\,|0\rangle\langle 0| + \frac{2}{3}\,|1\rangle\langle 1|)\ \mathsf{in}\ \{\{\!\{\lambda xy.x\}\!\}, \{\!\{\lambda xy.y\}\!\}\} \\ &= \mathsf{letcase}\ x = \{\!\{\pi^1(\frac{1}{3}\,|0\rangle\langle 0| + \frac{2}{3}\,|1\rangle\langle 1|)\}\!\}\ \mathsf{in}\ \{\{\!\{\lambda xy.x\}\!\}, \{\!\{\lambda xy.y\}\!\}\} \\ &= \{\!\{t\}\!\}\end{aligned}$$

Using the translations $\{\!\{\cdot\}\!\}$ and $(\!|\cdot|\!)$ (cf. Section 3.1), we can map any term in λ_ρ^o to a quantum closure in λ_q.

$$(\!|\cdot|\!) \circ \{\!\{\cdot\}\!\} : \Lambda_\rho^o \to \mathcal{C}_q$$

3.3.1 Correctness

We prove below that the translation of a well-typed term in λ_ρ^o preserves its type.

Theorem 3.3.4 (Type preservation) *If $t \in \Lambda_\rho^o$ and $\Gamma \vdash t : A$, then $\Gamma \vdash \{\!\{t\}\!\} : A$*

Proof *We proceed by induction on the derivation of $\Gamma \vdash t : A$.*

- *Let $\Delta, x : A \vdash x : A$ as a consequence of rule* ax*. By rule* ax*, we have $\Delta, x : A \vdash x : A$.*
- *Let $\Gamma \vdash \lambda x.t : A \multimap B$ as a consequence of $\Gamma, x : A \vdash t : B$ and rule $\multimap_i$. By the induction hypothesis $\Gamma, x : A \vdash \{\!\{t\}\!\} : B$. Then, by rule $\multimap_I$, we have $\{\!\{\Gamma\}\!\} \vdash \lambda x.\{\!\{t\}\!\} : A \multimap B$. Notice that $\lambda x.\{\!\{t\}\!\} = \{\!\{\lambda x.t\}\!\}$.*
- *Let $\Gamma, \Delta \vdash tr : B$ as a consequence of $\Gamma \vdash t : A \multimap B$, $\Delta \vdash r : A$ and rule $\multimap_e$. By the induction hypothesis, $\Gamma \vdash \{\!\{t\}\!\} : A \multimap B$, and $\Delta \vdash \{\!\{r\}\!\} : A$. Then, by rule $\multimap_e$, we have $\Gamma, \Delta \vdash \{\!\{t\}\!\}\{\!\{r\}\!\} : B$. Notice that $\{\!\{tr\}\!\} = \{\!\{t\}\!\}\{\!\{r\}\!\}$.*
- *Let $\Delta \vdash \rho^n : n$ as a consequence of rule ax_ρ. By rule ax_ρ, $\Delta \vdash \rho^n : n$. Notice that $\{\!\{\rho\}\!\}^n = \rho^n$*
- *Let $\Gamma \vdash U^m t : n$ as a consequence of $\Gamma \vdash t : n$ and rule* u*. By the induction hypothesis, $\Gamma \vdash \{\!\{t\}\!\} : n$. By rule* u*, we have $\Gamma \vdash U^m\{\!\{t\}\!\} : n$. Notice that $\{\!\{U^m t\}\!\} = U^m\{\!\{t\}\!\}$.*
- *Let $\Gamma \vdash \pi^m t : (m, n)$ as a consequence of $\Gamma \vdash t : n$ and rule* m*. By the induction hypothesis, $\Gamma \vdash \{\!\{t\}\!\} : n$. By rule* m*, we have $\Gamma \vdash \pi^m\{\!\{t\}\!\} : (m, n)$. Notice that $\{\!\{\pi^m t\}\!\} = \pi^m\{\!\{t\}\!\}$.*
- *Let $\Gamma, \Delta \vdash t \otimes r : n + m$ as a consequence of $\Gamma \vdash t : n$, $\Delta \vdash r : m$ and rule $\otimes$. By the induction hypothesis, $\Gamma \vdash \{\!\{t\}\!\} : n$, and $\Delta \vdash \{\!\{r\}\!\} : m$. Then, by rule $\otimes$, we have $\Gamma, \Delta \vdash \{\!\{t\}\!\} \otimes \{\!\{r\}\!\} : B$. Notice that $\{\!\{t \otimes r\}\!\} = \{\!\{t\}\!\} \otimes \{\!\{r\}\!\}$.*
- *Let $\Gamma, \Delta \vdash \mathsf{letcase}^o\ x = r\ \mathsf{in}\ t_1 \dots t_{2^n} : A$ as a consequence of $\Delta, x : n \vdash t_1 : A$, ..., $\Delta, x : n \vdash t_{2^n} : A$, and $\Gamma \vdash r : (m, n)$.*

  By the induction hypothesis, $\Delta, x : n \vdash \{\!\{t_1\}\!\} : A$, ..., $\Delta, x : n \vdash \{\!\{t_{2^n}\}\!\} : A$, and $\Gamma \vdash \{\!\{r\}\!\} : (m, n)$.

  Then, by rule lc*, we have $\Gamma, \Delta \vdash \mathsf{letcase}\ x = \{\!\{r\}\!\}\ \mathsf{in}\ \{\!\{t_1\}\!\} \dots \{\!\{t_{2^n}\}\!\} : A$.*

  Notice that $\mathsf{letcase}\ x = \{\!\{r\}\!\}\ \mathsf{in}\ \{\!\{t_1\}\!\} \dots \{\!\{t_{2^n}\}\!\} = \{\!\{\mathsf{letcase}^o\ x = r\ \mathsf{in}\ t_1 \dots t_{2^n}\}\!\}$.

- *Let* $\Gamma \vdash \sum_{i=1}^n p_i t_i : A$ *as a consequence of* $\Gamma \vdash t_1 : A, \dots, \Gamma \vdash t_1 : A$ *and rule* $+$. *By the induction hypothesis,* $\Gamma \vdash \{\!\{t_1\}\!\} : A, \dots, \Gamma \vdash \{\!\{t_n\}\!\} : A$. *Then we have the following type derivation.*

$$\dfrac{\dfrac{}{\Gamma, x : n \vdash t_1 : A}\,ih \quad \dots \quad \dfrac{}{\Gamma, x : n \vdash t_{2^k} : A}\,ih \quad \dfrac{\dfrac{}{\vdash \rho^k : n}\,\mathsf{ax}_\rho}{\vdash \pi^k\ \rho^k : (m,n)}\,\mathsf{m}}{\Gamma \vdash \mathsf{letcase}\ x = \pi^k\ \rho^k\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \dots, \{\!\{t_{2^k}\}\!\}\} : A}\,\mathsf{lc}$$

Where $t_{n+1} = \dots = t_{2^k} = t_1$. *Notice that* $\mathsf{letcase}\ x = \pi^k\ \rho^k\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \dots, \{\!\{t_{2^k}\}\!\}\} = \{\!\{\mathsf{letcase}\ = \pi^k\ \rho^k\ \mathsf{in}\ \{t_0, \dots, t_{2^m-1}\}\}\!\}$. $\square$

We also prove that the translation preserves the interpretation given in Table 2.6. This means that the translation of a term has the same meaning as the original term.

Theorem 3.3.5 *If* $t \in \Lambda_\rho^o$ *and* θ *is a valuation,* $[\![t]\!]_\theta = [\![\{\!\{t\}\!\}]\!]_\theta$.

Proof *We proceed by induction on* t.

Since the translation $\{\!\{\cdot\}\!\}$ *is shallow in the constructors shared by both calculi, the only interesting case is when* $t = \sum_{i=1}^n p_i\, t_i$. *By the induction hypothesis,* $[\![\{\!\{t_i\}\!\}]\!]_{\theta'} = [\![t_i]\!]_{\theta'}$ *for* $1 \le i \le n$.

Let $k = \lceil \log_2(n) \rceil$, $t_{n+1} = \dots = t_{2^k} = t_1$, *and* $\rho^k = \sum_{i=1}^n p_n\, |i\rangle\langle i|$. *Then,*

$$\begin{aligned}
[\![\{\!\{t\}\!\}]\!]_\theta &= \{(s_l\, q_{lj}, b'_{lj}, e_{lj}) \mid [\![\pi^k\ \rho^k]\!]_\theta = \{(s_l, b_l, \rho_l)\}_l, \text{ and} \\
&\qquad\qquad [\![\{\!\{t_{b_l}\}\!\}]\!]_{\theta, x=(\varepsilon,\rho_i)} = \{(q_{lj}, b'_{lj}, e_{lj})\}_j\} \\
&= \{(p_i\, q_{ij}, b'_{ij}, e_{ij}) \mid [\![t_i]\!]_{\theta, x=(\varepsilon,|i\rangle\langle i|)} = \{(q_{ij}, b'_{ij}, e_{ij})\}_j\}_{i=1}^n \\
&= [\![t]\!]_\theta
\end{aligned}$$

Notice that $\{\!\{t\}\!\} = \mathsf{letcase}\ x = \pi^k \rho^k\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \dots, \{\!\{t_{2^k}\}\!\}\}$, $[\![\pi^k \rho^k]\!]_\theta = \{(p_i, i, |i\rangle\langle i|)\}_{i=1}^n$, *and* $[\![t_i]\!]_{\theta, x=(\varepsilon,|i\rangle\langle i|)} = [\![t_i]\!]_\theta$ *since* $x \notin \mathsf{FV}(t_i)$. $\square$

Remark 3.3.6 *Unfortunately, the operational semantics is not preserved by the translation. Indeed, consider the terms*

$$t = (\lambda xy.x)(\frac{1}{3}\, t_1 + \frac{2}{3}\, t_2) \text{ and } r = \lambda y.(\frac{1}{3}\, t_1 + \frac{2}{3}\, t_2),$$

where $t \rightsquigarrow r$.

The translation of the terms have the following traces:

$$\begin{array}{c}
\{\!\{t\}\!\} = (\lambda xy.x)(\mathsf{letcase}\ x = \pi^1 \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{2}{3} \end{bmatrix}\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \{\!\{t_2\}\!\}\}) \\
\swarrow^{\frac{1}{3}} \qquad \searrow^{\frac{2}{3}} \\
\lambda y.\{\!\{t_1\}\!\} \qquad \lambda y.\{\!\{t_2\}\!\}
\end{array}$$

$$\{\!\{r\}\!\} = \lambda y.(\mathsf{letcase}\ x = \pi^1 \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{2}{3} \end{bmatrix}\ \mathsf{in}\ \{\{\!\{t_1\}\!\}, \{\!\{t_2\}\!\}\}) \in V_\rho$$

We cannot find a term or set of terms to close the translation and reduction diagram, because $\{\!\{t\}\!\}$ *and* $\{\!\{r\}\!\}$ *reduce to different sets of value terms. We could, however, prove that both sets of derived terms are indistinguishable. That is, that both represent the same quantum states in the underlying semantics.*

Remark 3.3.7 *We can solve the problem described in Remark 3.3.6 by modifying the reduction rules of λ_ρ^o and changing the reduction strategy to call-by-base (also called call-by-basis, cf. [DCGMV19]).*

This reduction strategy modifies the call-by-value strategy by propagating first the sum constructors to the summands of the term, before applying any β-reduction or reducing the $\mathsf{letcase}^o$ *construction. Since we make sure that the parameter is a non-sum term before reducing a function, this could solve the problem shown in the previous remark where a sum term was substituted inside a lambda abstraction.*

In order to use CBBase, we need to add the following rules to the calculus:

$$t\,(\sum_i p_i\,r_i) \rightsquigarrow \sum_i p_i\,(t\,r_i)$$

and

$$\mathsf{letcase}^o\ x = (\sum_i p_i\,r_i)\ \mathsf{in}\ \{t_1,\dots,t_n\} \rightsquigarrow \sum_i p_i\,\mathsf{letcase}^o\ x = r_i\ \mathsf{in}\ \{t_1,\dots,t_n\}$$

Since this modification requires altering the original calculus and rewriting a number of proofs from the original paper, we left this modification for a future work.

3.4 Pseudoinverse from λ_ρ to λ_ρ^o

As shown in Example 3.3.3 the translation $\{\!\{\cdot\}\!\}$ is not injective, and therefore it does not have an inverse. We can, however, define a pseudo inverse $\{\!\{\cdot\}\!\}^{-1} : Im(\{\!\{\cdot\}\!\}) \to \lambda_\rho^o$. That is, a function such that the composition $\{\!\{\cdot\}\!\} \circ \{\!\{\cdot\}\!\}^{-1} \circ \{\!\{\cdot\}\!\}$ equals $\{\!\{\cdot\}\!\}$.

Since the image of the translation $\{\!\{\cdot\}\!\}$ is the intersection between the terms of λ_ρ and λ_ρ^o, we define the pseudo inverse as the identity function, $\{\!\{\cdot\}\!\}^{-1} = \mathsf{id}$.

Below we prove that $\{\!\{\cdot\}\!\}^{-1}$ is effectively a pseudoinverse.

Lemma 3.4.1 *Let $t \in \Lambda_\rho^o$, then $\{\!\{t\}\!\} = \{\!\{\{\!\{\{\!\{t\}\!\}\}\!\}^{-1}\}\!\}$.*

Proof *Since $\{\!\{\cdot\}\!\}$ is shallow in the constructors shared by the calculi λ_ρ and λ_ρ^o, $\{\!\{\{\!\{\{\!\{t\}\!\}\}\!\}^{-1}\}\!\}$ $= \{\!\{\{\!\{t\}\!\}\}\!\}^{-1} = \{\!\{t\}\!\}$.* □

Chapter 4

Conclusions

In this thesis we have defined a translation between the quantum lambda calculi λ_ρ and λ_q. This translation required the encoding of mixed quantum states as bigger pure states using a purification method. This translation proved to be well formed, maintaining the operational semantics of the terms (Theorem 3.1.9). A direct corollary from this theorem is the strong normalization property of λ_ρ (Corollary 3.1.10).

While we were able to define the translation from λ_ρ to λ_q directly, it is not possible to define an inverse translation due to the inability to separate references to entangled qubits in λ_ρ. We instead defined a left-inverse for the previous translation.

We then defined a translation from λ_ρ^o to λ_ρ, by transforming the generalized density matrices of terms to terms that performed a non-deterministic choice by measuring a specially assembled quantum state. With this translation we mapped the terms of λ_ρ^o to the intersection between the terms of λ_q and of λ_ρ. This translation preserves the interpretation of the terms (Theorem 3.3.5), but it does not preserve the operational semantics. Since the translation's image is the intersection between the terms of λ_ρ and λ_ρ^o, we defined a pseudoinverse using the identity function.

Composing both translations, we finally obtained a translation from λ_ρ^o to λ_q.

4.1 Future work

As a future line of work, we want to modify the reduction rules of λ_ρ^o as mentioned in Remark 3.3.7 in order to prove that the translation from λ_ρ^o to λ_ρ preserves the operational semantics and prove the strong normalization of λ_ρ^o.

We also want to modify λ_ρ by adding a concept of local contexts with pointer variables in order to to allow for separating references to entangled qubits in a density matrix. This would allow us to define a complete translation from λ_q to λ_ρ.

Additionally, an alternative formulation of the translation from λ_ρ^o to λ_q could be defined, sending terms to sets of quantum closures in λ_q with associated probabilities. This may prove to be a more straightforward translation, though again it would not be possible to define a full inverse.

Bibliography

[AAKV01] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 50–59. ACM, 2001.

[ABN+01] Andris Ambainis, Eric Bach, Ashwin Nayak, Ashvin Vishwanath, and John Watrous. One-dimensional quantum walks. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 37–49. ACM, 2001.

[AD17] Pablo Arrighi and Gilles Dowek. Lineal: A linear-algebraic lambda-calculus. *Logical Methods in Computer Science*, 13(1:8), 2017.

[ADC12] Pablo Arrighi and Alejandro Díaz-Caro. A System F accounting for scalars. *Logical Methods in Computer Science*, 8(1:11), 2012.

[ADCP+14] Ali Assaf, Alejandro Díaz-Caro, Simon Perdrix, Christine Tasson, and Benoît Valiron. Call-by-value, call-by-name and the vectorial behaviour of the algebraic λ-calculus. *Logical Methods in Computer Science*, 10(4:8), 2014.

[ADCV17] Pablo Arrighi, Alejandro Díaz-Caro, and Benoît Valiron. The vectorial λ-calculus. *Information and Computation*, 254(1):105–139, 2017.

[AG05] Thorsten Altenkirch and Jonathan J. Grattage. A functional quantum programming language. In *Proceedings of LICS-2005*, pages 249–258. IEEE Computer Society, 2005.

[Bar84] H.P. Barendregt. *The lambda calculus: its syntax and semantics*. Studies in logic and the foundations of mathematics. North-Holland, 1984.

[BP15] Costin Bădescu and Prakash Panangaden. Quantum alternation: Prospects and problems. In Chris Heunen, Peter Selinger, and Jamie Vicary, editors, *Proceedings of QPL-2015*, volume 195 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–42, 2015.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940.

[DC17] Alejandro Díaz-Caro. A lambda calculus for density matrices with classical and probabilistic controls. In Bor-Yuh Evan Chang, editor, *Programming Languages and Systems (APLAS 2017)*, volume 10695 of *Lecture Notes in Computer Science*, pages 448–467. Springer, Cham, 2017.

[DCD17] Alejandro Díaz-Caro and Gilles Dowek. Typing quantum superpositions and measurements. `arXiv:1601.04294`, 2017.

[DCGMV19] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel, and Benoît Valiron. Realizability in the unitary sphere. *CoRR*, abs/1904.08785, 2019.

[DCP12] Alejandro Díaz-Caro and Barbara Petit. Linearity in the non-deterministic call-by-value setting. In Luke Ong and Ruy de Queiroz, editors, *Proceedings of WoLLIC 2012*, volume 7456 of *LNCS*, pages 216–231, 2012.

[DP06] Ellie D'Hondt and Prakash Panangaden. Quantum weakest preconditions. *Mathematical Structures in Computer Science*, 16:429–451, 2006.

[FDY11] Yuan Feng, Runyao Duan, and Mingsheng Ying. Bisimulation for quantum processes. *ACM SIGPLAN Notices (POPL'11)*, 46(1):523–534, 2011.

[FYY13] Yuan Feng, Nengkun Yu, and Mingsheng Ying. Model checking quantum markov chains. *Journal of Computer and System Sciences*, 79(7):1181–1198, 2013.

[GLR+13] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. *ACM SIGPLAN Notices (PLDI'13)*, 48(6):333–342, 2013.

[KC09] D.R. Kincaid and E.W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing.* American Mathematical Society, 2009.

[KLM07] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing.* Oxford University Press, Inc., New York, NY, USA, 2007.

[Kni96] Emanuel H. Knill. Conventions for quantum pseudocode. Technical Report LA-UR-96-2724, Los Alamos National Laboratory, 1996.

[Mer07] N. David Mermin. *Quantum Computer Science: An Introduction.* Cambridge University Press, 2007.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010.

[Plo75] G.D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(2):125 – 159, 1975.

[PSV14] Michele Pagani, Peter Selinger, , and Benoît Valiron. Applying quantitative semantics to higher-order quantum computing. *ACM SIGPLAN Notices (POPL'14)*, 49(1):647–658, 2014.

[Rom19] Lucas Romero. Confluence of λ_ρ. In *Master's thesis in preparation*, 2019.

[Sel04] Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[SV05] Peter Selinger and Benoît Valiron. A lambda calculus for quantum computation with classical control. In *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications*, TLCA'05, pages 354–368, Berlin, Heidelberg, 2005. Springer-Verlag.

[SV08] Peter Selinger and Benoît Valiron. On a fully abstract model for a quantum linear functional language. In *Proceedings of the 4th International Workshop on Quantum Programming Languages (QPL 2006)*, volume 210, pages 123–137, Amsterdam, The Netherlands, The Netherlands, July 2008. Elsevier Science Publishers B. V.

[Vau09] Lionel Vaux. The algebraic lambda calculus. *Mathematical Structures in Computer Science*, 19:1029–1059, 2009.

[vT04] André van Tonder. A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33:1109–1135, 2004.

[Yin11] Mingsheng Ying. Floyd–hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems*, 33(6):19:1–19:49, 2011.

[Yin16] Mingsheng Ying. *Foundations of Quantum Programming*. Elsevier, 2016.

[YYF12] Mingsheng Ying, Nengkun Yu, and Yuan Feng. Defining quantum control flow. `arXiv:1209.4379`, 2012.

[YYF14] Mingsheng Ying, Nengkun Yu, and Yuan Feng. Alternation in quantum programming: from superposition of data to superposition of programs. `arXiv:1402.5172`, 2014.

[YYW17] Mingsheng Ying, Shenggang Ying, and Xiaodi Wu. Invariants of quantum programs: characterisations and generation. *ACM SIGPLAN Notices (POPL'17)*, 52(1):818–832, 2017.

[Zor16] Margherita Zorzi. On quantum lambda calculi: a foundational perspective. *Mathematical Structures in Computer Science*, 26(7):1107–1195, 2016.