

MEASUREMENTS IN PROOF NETS
AS HIGHER-ORDER QUANTUM CIRCUITS
高階量子回路としての Proof Net における量子測定

by

Akira Yoshimizu
由水輝

A Master Thesis
修士論文

Submitted to
the Graduate School of the University of Tokyo
on February 3, 2014
in Partial Fulfillment of the Requirements
for the Degree of Master of Information Science and
Technology
in Computer Science

Thesis Supervisor: Ichiro Hasuo 蓮尾一郎
Lecturer of Computer Science

ABSTRACT

In this thesis, we give a definition of proof nets that can handle quantum computation, especially measurement, and Geometry of Interaction (GoI for short) semantics of them. We identify our proof nets as higher-order quantum circuits; by interpreting a quantum lambda calculus MLL_{qm} with measurement as our proof nets, we obtain a GoI semantics of the calculus as concrete token machines, i.e. in the form of automata.

Quantum computation has been drawing attention in recent years by its computational power. As well as developing efficient quantum algorithms, designing and studying quantum programming languages is also an attractive research topic. Linear logic has been used as type systems of such languages since it can suitably capture no-cloning theorem. However, linear logic itself is equipped with computation mechanisms, that are, proof nets and the GoI. Thus it is also possible to use it more directly, not as mere type systems. Such studies of utilizing linear logic proofs and the GoI in quantum setting are also done by researchers e.g. Dal Lago, Faggian, and Zorzi. We too aim at this direction of research.

Technically speaking, accommodating measurement is a major problem because of its inherent nondeterminacy. Our main contribution is that we newly introduce proof nets with quantum nodes that can handle measurement, which is the first one as far as we know. While the result is new, the design of the structure for accommodating measurement is much like that of traditional notions of weakening and additive conjunction. The reduction of our proof nets is shown to be strongly normalizing under a reasonable condition, which justifies the use of the nets as a quantum computation system. Terms of a quantum linear lambda calculus are interpreted as proof nets as which we regard higher-order version of quantum circuits. As an important example of quantum program, it can describe the quantum teleportation. We also give GoI semantics of the nets (and hence of the calculus), which moves multiple tokens at once and calculates probability along each slicing.

論文要旨

本研究は、測定を扱えるような量子 proof net の定義およびその Geometry of Interaction (GoI) 意味論を与え、それにより量子ラムダ計算の意味論を proof net への翻訳を介して与えるものである。

量子計算は近年注目を集めている研究分野の一つである。量子計算を用いた高速なアルゴリズムを設計する他、そのようなアルゴリズムを記述するための量子プログラミング言語の設計・研究もまた重要なトピックである。線型論理は量子力学における複製不可能定理に当たる性質を簡潔に表現できるという特徴があり、量子プログラミング言語では型システムに应用されることが多い。しかし、線型論理には proof net や Geometry of Interaction (GoI) という計算論的な道具立てが揃っており、単に型システムに組み込むのではなく線型論理をより直接的に意味論として用いることも可能である。この考え方のもと、proof net や GoI を量子計算的なセッティングのもとで研究することも Dal Lago, Faggian, Zorzi などによりなされており、我々の今回の研究もそれに連なるものである。

技術的には、測定を扱えるようにすることはその本質的な非決定性を扱わなくてはならないため大きな問題の一つである。我々の主要な貢献は、弱化および加法的論理積のための proof net の構造を援用することで測定を扱えるような量子 proof net およびその簡約規則を与えたことである。このアプローチにおいては測定を含むものは我々の知るかぎり初めてである。この proof net の簡約は量子計算として見て妥当な条件のもと強正規化性を持つため、量子計算を行う体系とみなすことができる。量子ラムダ計算の項は proof net として解釈される。さらに、加法的論理積の proof net の slicing の概念を用いることで proof net に GoI 意味論を与え、これにより proof net への翻訳を介することで測定を含む量子ラムダ計算の GoI による初の意味論を得た。

Acknowledgements

Thanks are due to Ugo Dal Lago, Kentaro Honda, Tristan Roussel, and Alexis Saurin for useful discussions and helpful comments. I am greatly indebted to Claudia Faggian, not only for suggesting valuable ideas but also for helping me to stay in Paris. Conversation with the members of Hasuo laboratory, especially the second year master students Toshiki Kataoka, Hiroyoshi Sekine, Kenta Cho, and Tetsuri Moriya, has always encouraged me. Finally, I would like to express the deepest gratitude to our supervisor Ichiro Hasuo.

This research is partially supported by International Exchange Dispatch Program in Graduate School of Information Science and Technology, The University of Tokyo.

Contents

1	Introduction	1
2	Preliminaries	4
2.1	Linear Logic	4
2.1.1	Sequent Calculus	4
2.1.2	MLL Proof Nets	5
2.1.3	Geometry of Interaction	7
2.2	Quantum Computation	10
3	Quantum Lambda Calculus MLLqm and MLLqm Proof Nets	12
3.1	Syntax of Quantum Lambda Calculus MLLqm	12
3.2	MLL Proof Nets with Quantum Nodes	13
3.2.1	MLLqm Proof Structures	13
3.2.2	Reduction of MLLqm Proof Structures	15
3.2.3	MLLqm Proof Nets and the Correctness Criterion	16
3.3	Translation of MLLqm Terms into Proof Nets	17
3.4	Examples and Discussion	20
4	Token Machine Semantics for MLLqm Proof Nets	22
4.1	Tokens	22
4.2	The Token Machine $\mathcal{T}_{\mathcal{N}}$	23
4.3	Big-Step Semantics of $\mathcal{T}_{\mathcal{N}}$	24
4.4	Soundness of the Token Machine Semantics	28
4.5	Example	29
5	Conclusions and Future Work	31
	References	32
A	Appendix	35
A.1	Proof of Lem. 3.5	35
A.2	Proof of Lem. 3.11	38
A.3	Proof of Lem. 3.13	38
A.4	Proof of Thm. 3.15	39
A.5	Proof of Lem. 4.5	40
A.6	Proof of Thm. 4.15	40

Chapter 1

Introduction

Quantum Programming Languages. Quantum computation and quantum communication have been attracting growing attention. The former achieves real breakthrough in computational power—at least for some classes of problems, such as the integer factorization problem (Shor’s algorithm [36]) and search problems (e.g. Grover’s algorithm [22]). While it is often disputed if quantum computation is physically realizable, quantum communication is close to actual deployment in real-world applications. By exploiting the nonlocal character of quantum phenomena (notably *quantum entanglement*), quantum cryptography protocols accomplish *perfect security* that do not rely on any computational assumptions (like Diffie-Hellman [8]).

Compared to the algorithmic aspects, the theory of *quantum programming* is relatively new. For example, quantum algorithms are most often expressed in *quantum circuits* that lack structuring means like recursion or higher-order functions. Consequently we have seen some proposals for quantum programming languages including QCL [32], quantum lambda calculi [35,38] and most recently Quipper [21]: QCL is imperative and the others are functional.

Our interests are in a quantum lambda calculus as a prototype of functional quantum programming languages. The functional style comes with several advantages. For one, a type system based on resource-sensitive *linear logic* [14] can force *no-cloning* of quantum states via type safety [38]. Moreover, various techniques for classical functional programming can often be “transferred” to the quantum setting, since they are formulated in an abstract mathematical language and hence are generic. For example, in [23,29,35] various semantical techniques in the classical setting—such as linear-nonlinear adjunctions, categorical geometry of interaction, and presheaf completion—are applied to quantum calculi, exploiting the categorical genericity of these techniques.

From Quantum Circuits to Proof Nets. The current work relies on another rich body of techniques that are developed in the linear logic community, among which are *proof nets* and *geometry of interaction*. Specifically we follow the line of [4,6] where, roughly speaking,

proof nets are thought of as *extended quantum circuits*.

Proof nets as devised in [14] can be seen as a graphical presentation of linear lambda terms (i.e. linear logic proofs) whose principal concern is reduction of terms (i.e. cut-elimination). Proof nets are “extended quantum circuits” in the following sense: (some) wires in proof nets can be naturally identified with those in quantum circuits; and at the same time higher-order computation is naturally accommodated using a linear type system (a function is typed by $A \multimap B \equiv$

$A^\perp \wp B$). This view is hence a quantum version of the one in [37]. See §3.4 for further discussion.

Once a quantum lambda term is presented as a proof net, the *geometry of interaction (GoI)* interpretation [16]—especially its concrete presentation as *token machines* [27]—gives a concrete and operational interpretation of the term as a state transition system. This is a main advantage of the current “proof net and GoI” approach compared to the categorical one taken in [23, 29]: in the latter models tend to be abstract and huge.

A main disadvantage, however, is that it is harder to interpret extra features in a calculus. Such desired features include recursion and accommodation of duplicable classical data by the ! modality; these are all present e.g. in [23]. In fact, in the preceding work [4, 6] of the current approach, even measurements are excluded from the calculi. Hence important (and basic) examples like quantum teleportation cannot be expressed in their calculi.

Contributions. In the current work we present a comprehensive framework for programming and interpreting higher-order quantum computation based on a linear lambda calculus, proof nets and GoI interpretation. More specifically:

- We introduce **MLLqm**, a linear lambda calculus with quantum primitives (including measurement, unlike [4, 6]).
- We define a notion of *proof net*, into which terms of **MLLqm** are translated. For accommodating measurements we follow the idea of (*additive*) *slices* (see e.g. [?]). We also define the reduction of proof nets and prove that it is strongly normalizing.
- We define *token machine semantics* of **MLLqm** proof nets and prove that it is *sound*, i.e., is invariant under reduction of proof nets. Here we have multiple tokens in a token machine (this is as in [6]); the slices are suitably handled following the token machine semantics in [26] for additives.

Our framework attains a balance between *expressivity* and *concreteness of models* that we find pleasant. On the one hand, the calculus **MLLqm** is reasonably expressive: it does include all the quantum operations (preparation, unitary transformation, and most importantly, measurement) and is capable of expressing examples like quantum teleportation, which is not possible in the earlier work [4, 6] of the same proof net approach. Moreover, our framework can naturally express higher-order procedures that are essential e.g. in formalizing *quantum pseudo-telepathy games* in quantum game theory. The latter are attracting attention as a useful presentation of quantum nonlocality (see e.g. [19]). On the other hand, while the languages in [23, 29, 35] are much more expressive—they include duplicable classical data (by the ! modality) and/or recursion—their models given in [23, 29] rely on abstract categorical constructions and it is not trivial to describe them in concrete terms. In contrast, our token machine semantics for **MLLqm** is given explicitly by a transition system.

The current work shares the same interest as [2], in the sense that both aim at pictorial formalisms for operational structures in quantum computation: 1) capture structures in procedures in quantum computation; and 2) allow—via rewriting or equational axioms—for reasoning on operational behaviors of quantum programs.

We follow the linear logic tradition; an advantage is explicit correspondence with a term calculus. In contrast, [2] employs string diagrams for monoidal

categories (more specifically compact closed categories with biproducts). The two approaches are not unrelated: there is a body of literature studying monoidal categories as models of linear logic. See [30] for a survey.

This thesis is based on the joint work with Claudia Faggian, Ugo Dal Lago, and our supervisor Ichiro Hasuo [40].

Organization of the Thesis. Preliminaries to read this thesis are provided in Chapter 2. After introducing the calculus MLLqm in §3.1, in §3.2 we define MLLqm *proof nets* and translate terms into proof nets. As usual, proof nets are defined to be *proof structures* satisfying a certain correctness criterion. We also define reduction (i.e. cut-elimination) of proof nets. In Chapter 4 we give GoI semantics to MLLqm proof nets, in the form of token machines. Our main result is soundness of the GoI semantics, i.e. that it is invariant under reduction of proof nets. Quantum teleportation will exemplify these constructions. Proofs of lemmas and theorems are deferred to Appendix if they are redundant or fairly long.

Chapter 2

Preliminaries

2.1 Linear Logic

Linear logic is a novel logical system devised by Girard in late 1980's [14]. It was originally developed from purely logical point of view. However, later linear logic (and its variants) turned out to be useful in other fields, namely in computer science. It is because linear logic can be viewed as “resource-sensitive” logic: intuitively, linear logic can distinguish those resources that can be used *only once* and those that can be used repeatedly. As one can imagine, this character allows finer analysis on computation in several topics in computer science, such as memory allocation and lock acquirement (e.g. [33, 39]). Moreover, linear logic is also useful in a new computation paradigm, that is, quantum computation. In this thesis we will exploit linear logic in the latter way.

2.1.1 Sequent Calculus

We assume familiarity to LK, i.e. the sequent calculus for classical logic. The syntactic equality shall be denoted by \equiv .

Syntactically speaking, (a fragment of) linear logic is one of *substructural logics*: it can be described as “logic without *weakening* and *contraction*”.

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \text{ WL} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \text{ CL}$$

Because of this prohibition, the following two groups of \wedge -introduction rules—which are equivalent in classical logic—are now *not* equivalent.

$$\boxed{\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} L\wedge_m \qquad \frac{\Gamma_1, \vdash \Delta_1, A \quad \Gamma_2, \vdash \Delta_2, B}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B} R\wedge_m}$$

$$\boxed{\frac{\Gamma, A \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} L\wedge_{a1} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} L\wedge_{a2} \quad \frac{\Gamma \vdash \Delta, A \quad \Gamma, \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B} R\wedge_a}$$

Hence, in this setting, these two groups yield *two* distinct conjunctions, called *multiplicative conjunction* \otimes and *additive conjunction* $\&$ respectively:

$$\boxed{\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta} L\otimes \qquad \frac{\Gamma_1, \vdash \Delta_1, A \quad \Gamma_2, \vdash \Delta_2, B}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \otimes B} R\otimes}$$

$$\boxed{\frac{\Gamma, A \vdash \Delta}{\Gamma, A \& B \vdash \Delta} L\&_1 \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \& B \vdash \Delta} L\&_2 \quad \frac{\Gamma \vdash \Delta, A \quad \Gamma, \vdash \Delta, B}{\Gamma \vdash \Delta, A \& B} R\&}$$

For the disjunction \vee the situation is similar; now there exist two disjunctions, *multiplicative disjunction* \wp and *additive disjunction* \oplus :

$$\frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \wp B} R\wp \quad \frac{\Gamma_1, A \vdash \Delta_1 \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \wp B \vdash \Delta_1, \Delta_2} L\wp$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \oplus B} R\oplus_1 \quad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \oplus B} R\oplus_2 \quad \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta} L\oplus$$

The axiom rule, exchange rule and cut rule are inherited from LK.

Linear negation $(\cdot)^\perp$ is then syntactically defined by de Morgan's law:

$$(A \otimes B)^\perp \equiv A^\perp \wp B^\perp, \quad (A \wp B)^\perp \equiv A^\perp \otimes B^\perp,$$

and introduced by the following rules:

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, A^\perp \vdash \Delta} L\perp \quad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, A^\perp} R\perp.$$

A logical formula A^\perp is also called the *dual* of A . Note that the multiplicative connectives (\otimes and \wp) are dual of each other, and so are the additive connectives ($\&$ and \oplus). The fragment consisting of multiplicative connectives only is called *multiplicative linear logic (MLL)*, while that consisting of both multiplicative ones and additive ones (but without *exponential modalities* described later) is called *multiplicative additive linear logic (MALL)*. MLL is known to behave especially well and we will mainly make use of this fragment.

Remark 2.1. Applying $R\perp$ rule repeatedly, we can move all formulas to the right-hand side of \vdash . Therefore it suffices to consider the right-hand side rules only. In the next subsection we use this one-sided sequent calculus.

Linear implication \multimap is defined by $A \multimap B \equiv A^\perp \wp B$. Intuitively, $A \multimap B$ is the type of a function that uses its argument of type A *exactly once* to produce a return value of type B .

Because of such linearity, multiple use of values (or ignorance of values) cannot be expressed by the connectives we have seen so far. It is recovered by two modalities, $!$ (read as *of course* or *bang*) modality and its dual $?$ (read as *why not*) modality, with identification $(!A)^\perp \equiv ?(A^\perp)$ and $(?A)^\perp \equiv !(A^\perp)$.

Intuitively, a formula $!A$ describes “possibility to produce as much A 's as you like”. Introduction rules for $!$ modality are as follows (since $?$ modality is dual of $!$ modality, we omit the rule for $?$ modality). Note that they include modified version of weakening and contraction ($W!$ and $C!$ respectively).

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta} W! \quad \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta} C! \quad \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} D! \quad \frac{! \Gamma, A \vdash ? \Delta}{! \Gamma, !A \vdash ? \Delta} P!$$

Denotational semantics of linear logic can be given by *coherent semantics*, or more abstractly by *linear-nonlinear adjunctions* in category theory. Since we do not utilize them in this thesis, we just suggest some pointers: for non-categorical ones Girard's original paper [14], or the appendix of [9]. For categorical one [30] is a comprehensive survey.

2.1.2 MLL Proof Nets

Proof nets were developed also by Girard in [14]. They are a representation of linear logic proofs as certain kind of directed graphs, often quoted as “graphical syntax” of linear logic proofs. Proof nets not only allow us to handle proofs

graphically, but also abstract more essential part of proofs like natural deduction does so to intuitionistic logic.

For the multiplicative fragment MLL, proof nets behave especially well. It is also possible to accommodate exponential part appropriately, but we only offer the reader a reference [34] since we do not deal with it in the thesis.

First we define *proof structures* as follows:

Definition 2.2 (MLL proof structure). An (MLL) *proof structure* \mathcal{S} is a directed graph consisting of nodes in Fig. 2.1, satisfying the following two conditions:

- each edge in \mathcal{S} is well-typed, and
- no incoming edge in \mathcal{S} is dangling.

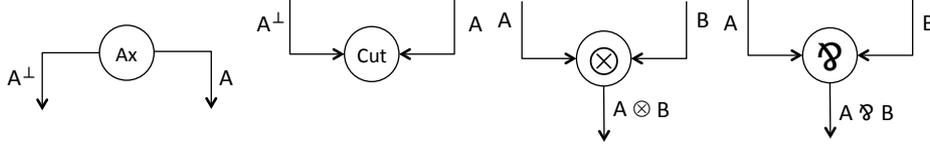


Figure 2.1: MLL nodes

Roughly speaking, a *proof net* is a proof structure that corresponds to (possibly many) linear sequent calculus proof(s). Indeed, we can translate sequent calculus proofs to proof structures. We define a translation $\overline{(\cdot)} : \{\text{sequent calculus proofs}\} \rightarrow \{\text{proof structures}\}$ as follows:

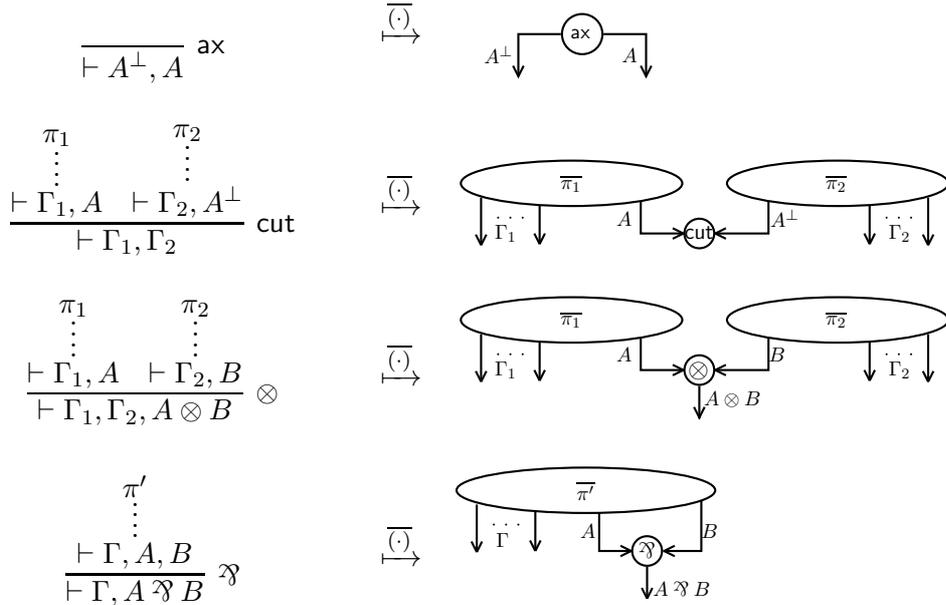
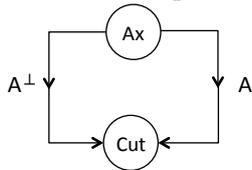


Figure 2.2: translation from sequent calculus to proof structure

To define proof nets as those proof structures contained in the image of $\overline{(\cdot)}$ appears possible. However, it can be immediately checked that not all proof structures come from sequent calculus proofs. Typical one is shown below.



What allows us to distinguish such meaningless structures and valid ones is characterization of proof nets, called *correctness criterion*. Girard first gave a correctness criterion using the notion of trip [14], and currently Danos and Regnier's

“connected and acyclic” one [7] is widely used in the linear logic community. We too use a variant of Danos and Regnier’s one later.

Definition 2.3 (Danos and Regnier’s correctness criterion). Let \mathcal{S} be a proof structure. A *correctness graph* of \mathcal{S} is an undirected graph obtained by applying the following operations to \mathcal{S} .

- Ignore directions of all edges.
- For each \wp node, choose one of the two premises and disconnect the other.
- The other nodes (ax, cut, \otimes) are not changed.

A correctness graph is said to *satisfy the correctness criterion* if it is acyclic and connected.

Observe that the correctness graph of the example above is cyclic and thus does not satisfy the correctness criterion. Indeed Lem. 2.5 below holds and we define proof nets to be those proof structures satisfying the correctness criterion.

Definition 2.4 (Proof net). A proof structure \mathcal{S} is called a *proof net* if each of its correctness graphs satisfies the correctness criterion.

Lemma 2.5. *Let \mathcal{S} be a proof structure. Then \mathcal{S} satisfies the correctness criterion if and only if there exists a linear sequent calculus proof π s.t. $\bar{\pi} = \mathcal{S}$.*

A notion of *reduction* of proof nets corresponds to that of natural deduction.

Definition 2.6 (Reduction rules). *Reduction rules* are shown in Fig. 2.3. The first one erases the cut node directly connected to the ax node. The latter one reduces size of formulas, erasing the \otimes node and the \wp node.

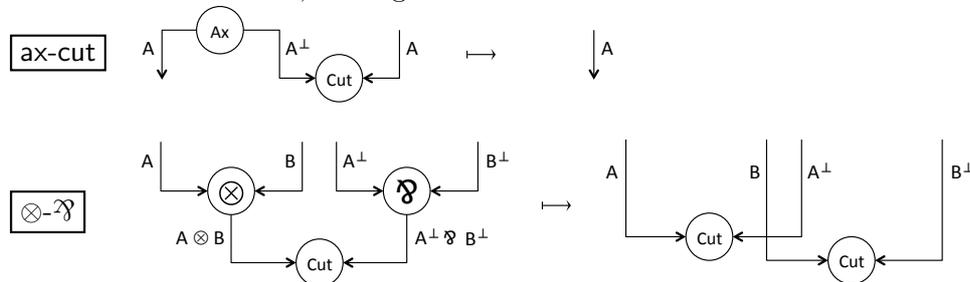


Figure 2.3: reduction rules

Reduction of proof structures maps a proof net to another proof net, i.e. preserves the correctness criterion. It is also strongly normalizing and confluent. Thus it can be seen as an equivalent of normalization of proofs.

Remark 2.7. Like natural deduction, proof nets can encode terms of lambda calculus. Especially, *linear* lambda calculus (each bound variable is used *exactly once* in its body) can be encoded using MLL proof nets only. We extend this encoding to a quantum calculus in this thesis. Encodings of both linear and full (i.e. not necessarily linear) lambda calculus are described in [34] in detail.

2.1.3 Geometry of Interaction

The *geometry of interaction*, or *GoI* in short, is a dynamic semantics of linear logic proofs. Initially it was introduced (once again by Girard [16] and his succeeding papers [15,17]) in order to analyze the cut elimination procedure of linear

logic in a more essential way. In his original paper, the semantics of a linear logic proof is represented as matrixes of operators in C^* -algebra, and the *execution formula* is calculated from the matrixes. This formula is shown to be invariant under cut elimination. Then the strong normalization property of cut elimination is mathematically assured, namely by the nilpotency of a subformula of the execution formula.

The matrixes can be considered as an encoding of the data of “where information came from and will go to?”. Though this C^* -algebra representation works well, it is a bit too sophisticated to represent it. Indeed, there has been proposed more elementary and concrete representations that are essentially the same as the original one. Among them, we exploit *token machine semantics*, which describes the data as concrete transition systems as in [26]. Intuitively, in this semantics a particle, or *token*, starts from a conclusion of a proof net and circulates around the net according to several simple rules. Then the semantics of the net is given by the data of “from which conclusion to which conclusion the particle moves.” Here we briefly see the semantics for MLL.

First we see the definition of *context*, which designates an atom occurring in a formula. In other words, it tells by which ax node the atom is introduced.

Definition 2.8 (Context). A *context* is defined by the following BNF:

$$C ::= [] \mid C \otimes A \mid A \otimes C \mid C \wp A \mid A \wp C ,$$

where A is a logical formula of MLL. Note that every context has exactly one *hole* $[]$. The type obtained by substituting a type A for the hole in a context C is denoted by $C[A]$. A context C is called a *context for* A if the type A is obtained by substituting some type B for the hole $[]$, i.e. $A \equiv C[B]$. The linear negation C^\perp of a context C is defined in a natural way, e.g. $(A \otimes [])^\perp := A^\perp \wp []$.

Using the notion of context, a *state* describes “where a token is and which direction it goes.”

Definition 2.9 (State). Given a proof net \mathcal{S} , a *state* of the token machine $\mathcal{T}_{\mathcal{S}}$ is a triple (A, C, D) where

- A is an edge of \mathcal{S} , represented by the type occurrence A assigned to it,
- C is a context for A , and
- D is a *direction*, that is an element of $\{\uparrow, \downarrow\}$.

Definition 2.10 (Transition $\rightarrow_{\mathcal{S}}$ of the token machine $\mathcal{T}_{\mathcal{S}}$). The *transition relation* $\rightarrow_{\mathcal{S}}$ between states of the token machine $\mathcal{T}_{\mathcal{S}}$ is defined by the rules as in Fig. 2.4.

For each rule, we informally depict the intended movement of the token.

Then *initial states* and *final states* are defined as follows. A token starts from an initial state and will go into a final state.

Definition 2.11 (Initial states and final states). Let \mathcal{S} be an MLL proof net. A state (A, C, D) is said to be *initial* if A is a conclusion edge, $C[\alpha] \equiv A$ for some atomic type α , and $D = \uparrow$; said to be *final* if A is a conclusion edge, $C[\alpha] \equiv A$ for some atomic type α , and $D = \downarrow$.

The set of initial states and that of final states are denoted by $I_{\mathcal{S}}$ and $F_{\mathcal{S}}$, respectively.

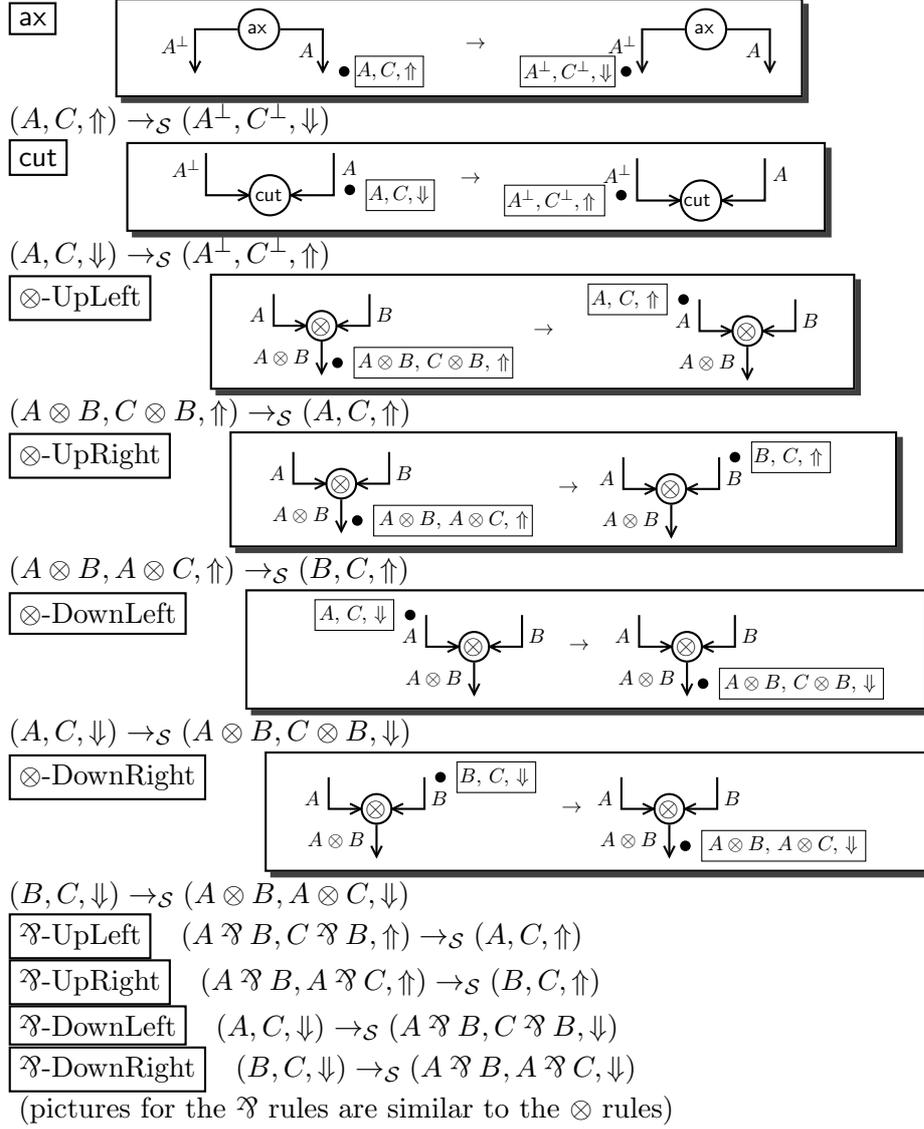


Figure 2.4: transition rules for $\mathcal{T}_{\mathcal{S}}$

Finally we define the semantics of MLL proof nets.

Definition 2.12 (Token machine semantics). Let \mathcal{S} be an MLL proof net. The *token machine semantics* of the \mathcal{S} , denoted by $\llbracket \mathcal{S} \rrbracket$, is the function $\llbracket \mathcal{S} \rrbracket : I_{\mathcal{S}} \rightarrow F_{\mathcal{S}}$ defined by $\llbracket \mathcal{S} \rrbracket (s) := s' \in F_{\mathcal{S}}$ if $s \rightarrow_{\mathcal{S}}^{\dagger} s'$.

This semantics is proved to be invariant under reduction of proof nets. In other words, the transition rules shows those paths that are preserved by reduction. Here we would like to make two remarks:

Remark 2.13. On an MLL proof net (and also on an MELLL proof net) one token circulates along a net at a time. In [6] and this thesis, a modification is made on this characteristic: there will be (possibly) more than one token at once.

Remark 2.14. The invariance of the semantics under reduction means that, if a proof net translated from a lambda calculus term is given, GoI can calculate the result of its normal form without (β -)reduction. In [27] and [34], this aspect is employed to calculate—as concrete implementation of compilers—the result of

terms of lambda calculus extended with natural numbers, i.e. PCF. In this thesis we apply similar extension to quantum lambda calculus.

In [34], GoI semantics is described by using so-called *path algebra*. Instead of actually traversing a proof net via transition system, this representation calculates an algebraic term along paths on a proof net. Then the valid paths (called *persistent paths* there) are defined as such paths that is not algebraically equal to 0. Persistent paths are precisely the paths tokens will go along in token machine semantics.

Since there is well-known Curry-Howard isomorphism between β -reduction and cut elimination, it is not surprising that GoI can be utilized in the study of computation. Such applications include compiler construction for functional programs [27,34], analysis of optimal reduction [20], and compilation of hardware description (a series of papers by Ghica et. al. [10–13]).

2.2 Quantum Computation

Quantum computation is a new paradigm of computation that uses quantum mechanics as computational devices. Its substance lies in the potential computational power. In 1994, Shor showed that the integer factoring problem can be solved using quantum computer, faster than any existence non-quantum algorithm (Shor’s algorithm [36]). Since this discovery, eager investigation for such *quantum algorithms* has been done.

In this section, we briefly see how notions of quantum computation can be discussed mathematically. For more explanation, a standard textbook in this area is [31]. Discussion from more mathematical viewpoint can be found in [24].

In general setting, quantum computation is usually written in the language of Hilbert space. If restricted to finite-dimensional case, a Hilbert space is isomorphic to the vector space \mathbb{C}^{2^n} , where n is the dimension of the space. In the following we only consider finite-dimensional case, hence \mathbb{C}^{2^n} suffices. An alternative representation called *density operator* is also useful, but in the scope of this thesis we do not employ it. For details see [31, §2.4].

Firstly, a state of a single *quantum bit*, called *qubit*, is represented by a normalized vector in the vector space \mathbb{C}^2 . The basis states of \mathbb{C}^2 are usually written as $|0\rangle$ and $|1\rangle$, respectively (Dirac’s notation). For instance, one of the basis ($|0\rangle$ or $|1\rangle$) represents a state of qubit: this state can be thought of as a usual bit being 0 or 1. A significant character of quantum computation is that a qubit can take a state in *superposition* of 0 and 1, for instance $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ which is neither $|0\rangle$ nor $|1\rangle$. Considering the normalization requirement, one state is generally described as $\alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}^2$ and $|\alpha|^2 + |\beta|^2 = 1$.

A state of a multiple-qubit system is then described as the tensor product of 1-qubit systems, i.e. the tensor product of vector spaces \mathbb{C}^2 . For example, 2-qubit system has 4 basis vectors $|0\rangle \otimes |0\rangle$, $|0\rangle \otimes |1\rangle$, $|1\rangle \otimes |0\rangle$, and $|1\rangle \otimes |1\rangle$. Such vectors are often written as $|j\rangle|k\rangle$ or more shortly $|jk\rangle$ (where $j, k \in \{0, 1\}$).

In multiple-qubit setting, one of the most significant features of quantum computation arises. As an example, consider the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. With straightforward calculation, it turns out that this state cannot be rewritten in the form of $|\varphi\rangle \otimes |\psi\rangle$ for any $|\varphi\rangle, |\psi\rangle \in \mathbb{C}^2$, which is not a possible situation in classical (i.e. non-quantum) computation. Such a phenomenon is called *quantum entanglement* or just *entanglement*, and the decent computational power of quantum computation (somehow) comes from this phenomenon. A state in entanglement is also called an *entangled state*.

The quantum counterpart of logical operation in classical computation is *unitary transformation*. Since a state of qubits is represented by a normalized vector in \mathbb{C}^{2^n} , modification of the state is written by 2^n -dimensional square matrixes. Considering the normalization requirement, it can be checked that such matrixes U must be *unitary*, i.e. $U^\dagger U = I$. In turn, any unitary matrix of dimension 2^n is indeed an operation on n qubits. Unitary transformations are often called *unitary gates*, since they correspond to logical gates in classical computation. Among such transformations, **CNOT** (*controlled not gate*) is particularly important one. As a matrix, it is defined to be

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

for basis vectors $|00\rangle, |01\rangle, |10\rangle$, and $|11\rangle$ in this order. It makes the first qubit unchanged and flips the second qubit if the first qubit is 1, with coefficients remaining the same. It is known that **CNOT** and 1-qubit unitary transformations consist the *universal set* of gates, in the sense that *any* unitary gate can be composed from elements of the set. **CNOT** gate also plays a crucial role in an important procedure, namely quantum teleportation.

Finally, *measurement* is a procedure to extract information from a state. (Here we only mention the *measurement of a qubit in the computational basis*. For other forms of measurement see [31, §2.2.3].) Actually, we cannot directly know the coefficients α and β of a state $\alpha|0\rangle + \beta|1\rangle$. Instead, we can only *measure* the qubit and obtain 0 or 1 according to α and β : 0 is obtained with probability $|\alpha|^2$ and 1 is obtained with probability $|\beta|^2$. Moreover, after measurement the state of qubits is no longer the same as the original one. If 0 is obtained by measuring $\alpha|0\rangle + \beta|1\rangle$, then the state is now $|0\rangle$ and we cannot restore it. In general, a state $|\varphi_0\rangle + |\varphi_1\rangle$ changes into $|\varphi'_0\rangle$ (resp. $|\varphi'_1\rangle$) if 0 (resp. 1) is measured, where

$$\begin{aligned} |\varphi_0\rangle &= \sum_j \alpha_j (|\psi_j^0\rangle \otimes |0\rangle \otimes |\chi_j^0\rangle), & |\varphi'_0\rangle &= \sum_j \frac{\alpha_j}{\sqrt{\sum_k |\alpha_k|^2}} (|\psi_j^0\rangle \otimes |0\rangle \otimes |\chi_j^0\rangle), \\ |\varphi_1\rangle &= \sum_j \beta_j (|\psi_j^1\rangle \otimes |1\rangle \otimes |\chi_j^1\rangle), & |\varphi'_1\rangle &= \sum_j \frac{\beta_j}{\sqrt{\sum_k |\beta_k|^2}} (|\psi_j^1\rangle \otimes |1\rangle \otimes |\chi_j^1\rangle), \end{aligned}$$

where $|\psi_j^b\rangle$ is of length $m - 1$ and m is the index of the qubit that is measured.

Chapter 3

Quantum Lambda Calculus MLLqm and MLLqm Proof Nets

3.1 Syntax of Quantum Lambda Calculus MLLqm

We introduce a typed calculus MLLqm. It is a term calculus based on linear logic—specifically *multiplicative linear logic (MLL)* that has connectives \otimes , \wp and $(\cdot)^\perp$. It is further augmented with quantum primitives that are rich enough to express any *quantum operation*. The latter notion is roughly for “what we can do to quantum states” and can be represented as a combination of *preparation*, *unitary transformation* and *measurement*. See §2.2 for brief introduction to these three primitives of quantum computation and [31, Chap. 8] for more details. The name MLLqm stands for “MLL for quantum computation with measurements.”

Definition 3.1 (Types of MLLqm). *Types* of MLLqm are defined by the following BNF:

$$A, B ::= \text{qbit} \mid \text{qbit}^\perp \mid A \otimes B \mid A \wp B.$$

The types qbit and its *dual* qbit^\perp are atomic. As is customary in linear logic, we syntactically identify types according to the following rules: $(A \otimes B)^\perp \equiv A^\perp \wp B^\perp$, $(A \wp B)^\perp \equiv A^\perp \otimes B^\perp$, and $(A^\perp)^\perp \equiv A$. We write $A \multimap B$ for $A^\perp \wp B$ and $A^{\otimes n}$ for $(\cdots (A \otimes A) \otimes A) \cdots \otimes A$ (here \otimes occurs $n - 1$ times).

Definition 3.2 (Terms of MLLqm). *Terms* of MLLqm are defined by:

$$\begin{aligned} M, N, L ::= & \quad x \mid \lambda x^A.M \mid MN \mid \langle M, N \rangle \mid \lambda \langle x^A, y^B \rangle.M \\ & \quad \mid \text{new}_{|\varphi\rangle} \mid \mathbf{U} \mid \text{if meas } M \text{ then } N \text{ else } L . \end{aligned}$$

Here x is an element of a fixed countable set \mathbf{Var} of *variables*. $\text{new}_{|\varphi\rangle}$ is a constant for each normalized vector $|\varphi\rangle$ in \mathbb{C}^2 and designates preparation of a qubit. \mathbf{U} is a constant for each 2^n -dimension unitary matrix, where $n \in \mathbb{N}$. Measurements meas occur only in conditionals. Note that in variable binders λx^A and $\lambda \langle x^A, y^B \rangle$, variables x, y come with explicit type labels. This is to ensure Lem. 3.5.

Remark 3.3. The constructor $\text{if meas } M \text{ then } N \text{ else } L$ is intended for “classical control”: operationally, the qubit represented by M is *actually measured* before going on to evaluate N or L .

This is not to be confused with “quantum control.” In quantum circuits, it is well-known that any measurement can be postponed to the end of a circuit (the *principle of deferred measurement*, [31, §4.4]). This is possible by use of *controlled operations* like CNOT [31, §4.3]. We shall stick to classical control because, in the current higher-order setting, it is not clear how to simulate classical control by quantum control, or how to systematically construct quantum controlled operations.

Definition 3.4 (Typing rules of MLLqm). Typing rules of MLLqm are shown below. A *context* Γ in a type judgment is a set $\{x_1 : A_1, \dots, x_n : A_n\}$ of variables and their types. We write its *domain* $\{x_1, \dots, x_n\}$ as $|\Gamma|$. The juxtaposition Γ, Δ of contexts denotes their union and we assume $|\Gamma| \cap |\Delta| = \emptyset$.

$$\begin{array}{c}
\frac{}{x : A \vdash x : A} \text{ ax} \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A. M : A \multimap B} \multimap I_1 \quad \frac{\Gamma, x : A, y : B \vdash M : C}{\Gamma \vdash \lambda \langle x^A, y^B \rangle. M : A \otimes B \multimap C} \multimap I_2 \\
\frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B} \multimap E \quad \frac{\Gamma \vdash M : A \quad \Delta \vdash N : B}{\Gamma, \Delta \vdash \langle M, N \rangle : A \otimes B} \otimes I \\
\frac{}{\vdash \text{new}_{|\varphi\rangle} : \text{qbit}} \text{ new} \quad \frac{}{\vdash U : \text{qbit}^{\otimes n} \multimap \text{qbit}^{\otimes n}} U_n \\
\frac{\Gamma \vdash M : \text{qbit} \quad \Delta \vdash N : A \quad \Delta \vdash L : A}{\Gamma, \Delta \vdash \text{if meas } M \text{ then } N \text{ else } L : A} \text{ meas}
\end{array}$$

The rule $\multimap I_2$ replaces the usual $\otimes E$ rule that is problematic in the current linear setting. As a profit of requiring variable binders to carry their types, the next lemma holds. It will enable inductive translation of terms into proof nets.

Lemma 3.5. *A derivable type judgment $\Gamma \vdash M : A$ has a unique derivation.*

Proof. By structural induction on M . Note that the typing rules are syntax-directed, i.e. the rule right above a judgment $\Gamma \vdash M : A$ in its derivation is uniquely determined by the structure of M . For details see Appendix. \square

3.2 MLL Proof Nets with Quantum Nodes

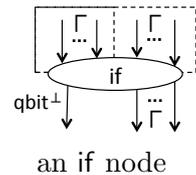
In this section we introduce the notion of proof net tailored for the calculus MLLqm. It is based on MLL proof nets [14] (see also [34]) and has additional nodes that correspond to quantum primitives (preparation, unitary transformation and measurement). Among them, (conditionals based on) measurements are the most challenging to model; we follow the idea of *additive slices* that are successfully utilized e.g. in [28].

As usual, we start with the notion of *proof structures* as graphs consisting of certain nodes. Then *proof nets* are defined to be those proof structures which comply with a *correctness criterion* (like Danos & Regnier's in [7]). We define translation of MLLqm terms into proof structures, which we prove to be proof nets. Moreover, we define reduction of proof structures, which we think of as one operational semantics of MLLqm terms. It is shown that proof nets are reduced to proof nets, and that reduction of proof nets is strongly normalizing (SN). Note that recursion is not in MLLqm.

3.2.1 MLLqm Proof Structures

Like the standard translation of λ calculus into MLL proof nets, we will interpret our calculus MLLqm by a graph structure. The difficulty lies in how to interpret measurement, and we accomplished it by a design inspired by the notions of weakening and additive slice.

In addition to the usual nodes in MLL proof nets, we introduce three kinds of nodes for quantum computation: *new* (preparation of a single qubit), *U* (unitary transformations/gates), and *if* (conditionals according to measurement of a qubit). An *if* node is as shown on the right. It is like a *box* in standard proof nets.



An if node will appear in a proof structure in the form where the two dashed boxes on its top are filled with “internal” proof structures. Such a combination of an if node and two (internal) proof structures shall be called a *meas node*. Overall, in MLLqm proof structures we allow the following seven kinds of nodes (Fig. 3.1).

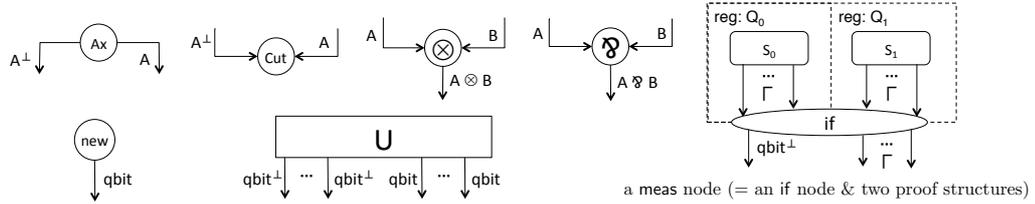


Figure 3.1: nodes of MLLqm proof structures

Note that nodes and proof structures are defined by mutual induction: in a proof structure there is a *meas* node, in whose dashed boxes there are other internal proof structures, and so on. We will make this precise in Def. 3.6. In Fig. 3.1, a unitary gate node for a 2^n -dimension unitary matrix U has n -many *qbit* edges and n -many *qbit* $^\perp$ edges. Γ denotes a finite sequence of types. In a *meas* node, the *qbit* $^\perp$ -typed edge sticking out to the down-left is called a *query edge*.

As usual, incoming edges of a node are called *premises* and outgoing edges are called *conclusions*. A proof structure is roughly a graph that consists of nodes in Fig. 3.1, and is augmented with a quantum state called a *quantum register*, whose functionality we shall explain by an example.

See Fig. 3.2. The outermost proof structure (we say it is of *level 0*) has

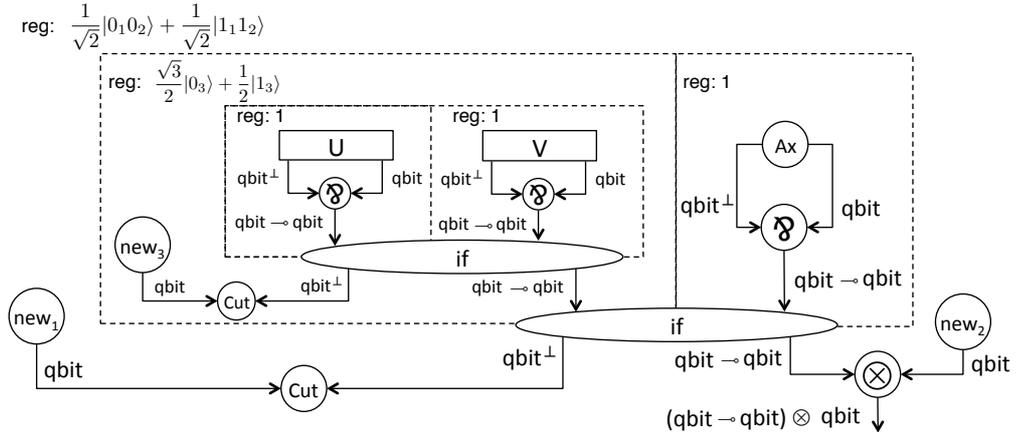


Figure 3.2: an example of proof structure

two new nodes, a cut node, a \otimes node and a *meas* node. Its quantum register is a state of a 2-qubit system; each qubit corresponds to a certain new node and the correspondence is designated by indices. Therefore our intention is that each proof structure has a quantum register whose size is the number of new nodes, and that the proof structure explicitly carries the content of the quantum register. Such pairing of computational structure (proof structures here) and quantum registers is inspired by the operational semantics of [35], where a term of a calculus and a quantum state together form a quantum closure.

Definition 3.6 (MLLqm proof structure). Let \mathcal{S} be a directed finite graph consisting of nodes in Fig. 3.1; Q be a quantum register of length $n \in \mathbb{N}$ (that is, a normalized vector in \mathbb{C}^{2^n}); k be the number of new nodes in \mathcal{S} ; and l be a bijection $\{\text{the new nodes in } \mathcal{S}\} \cong \{1, 2, \dots, k\}$. A triple (\mathcal{S}, Q, l) satisfying the following three conditions

- each edge in \mathcal{S} is well-typed;
- no incoming edge in \mathcal{S} is dangling; and
- $n = k$

is called a *proof structure*. The types on the dangling outgoing edges in \mathcal{S} are called the *conclusions* of \mathcal{S} .

Let $(\mathcal{S}_0, Q_0, l_0)$ and $(\mathcal{S}_1, Q_1, l_1)$ be proof structures with the same conclusions, say Γ . We call a triple (if node, $(\mathcal{S}_0, Q_0, l_0), (\mathcal{S}_1, Q_1, l_1)$) a *meas node* and regard it as a node with conclusions $\text{qbit}^\perp, \Gamma$. Each of the proof structures $(\mathcal{S}_0, Q_0, l_0)$ and $(\mathcal{S}_1, Q_1, l_1)$ is called a *branch* of the *meas node*.

The outermost proof structure is said to be *of level 0* and the branches of a *meas node* of level n are said to be *of level $n + 1$* .

We emphasize again that the above definitions of proof structures and *meas nodes* are mutually inductive. We allow *meas nodes* nested only finitely many times. The bijection l in a proof structure (\mathcal{S}, Q, l) gives indices to *new nodes* and designates correspondences between *new nodes* and qubits in a quantum register Q .

For example, in Fig. 3.2 the unitary gate nodes U and V belong to level 2. The quantum state that corresponds to the node new_3 is in the level-1 register. Note that it is invisible from level 0.

Finally we define *slices* for MLLqm proof structures, like usual additive slices. We will employ this notion later in §4.

Definition 3.7 (Slicing and slices). Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof structure. A *slicing* is a function $b : \{\text{all if nodes in } \mathcal{S} \text{ (of any level)}\} \rightarrow \{0, 1\}$. Abusing notation, a *slice* $b(\mathcal{N})$ is a graph obtained by deleting the unselected branch of each if node according to the slicing b , i.e. if $b(v) = 0$ delete the branch on the right and if $b(v) = 1$ delete the branch on the left for each if node v . Note that a slice is not a proof structure.

3.2.2 Reduction of MLLqm Proof Structures

We now introduce reduction rules for MLLqm proof structures. Following the Curry-Howard intuition that normalization of a proof is computation, a reduction step is thought of as a step in quantum computation.

Definition 3.8 (Reduction rules of MLLqm proof structures). *Reduction rules* are shown in Fig. 3.3. The first two are standard in MLL proof nets; the latter three are new. In the unitary gate rule, the unitary matrix U^{j_1, \dots, j_m} acts on j_1, \dots, j_m -th qubits in the same way as U does, and leaves other qubits unchanged. The last two rules occur probabilistically, where the resulting quantum registers $|\varphi'_0\rangle, |\varphi'_1\rangle$ and probabilities $\sum_j |\alpha_j|^2, \sum_j |\beta_j|^2$ defined in the usual way in quantum information theory. Explicitly:

$$\begin{aligned}
 |\varphi_0\rangle &= \sum_j \alpha_j (|\psi_j^0\rangle \otimes |0\rangle \otimes |\chi_j^0\rangle), & |\varphi'_0\rangle &= \sum_j \frac{\alpha_j}{\sqrt{\sum_k |\alpha_k|^2}} (|\psi_j^0\rangle \otimes |\chi_j^0\rangle), \\
 |\varphi_1\rangle &= \sum_j \beta_j (|\psi_j^1\rangle \otimes |1\rangle \otimes |\chi_j^1\rangle), & |\varphi'_1\rangle &= \sum_j \frac{\beta_j}{\sqrt{\sum_k |\beta_k|^2}} (|\psi_j^1\rangle \otimes |\chi_j^1\rangle),
 \end{aligned} \tag{1}$$

where $|\psi_j^b\rangle$ of length $m - 1$ and m is the index of the *new node* that is measured (here we ignore the measured qubit after measurement). The other rules occur with probability 1. In *meas rules*, the indexing function l is suitably updated too.

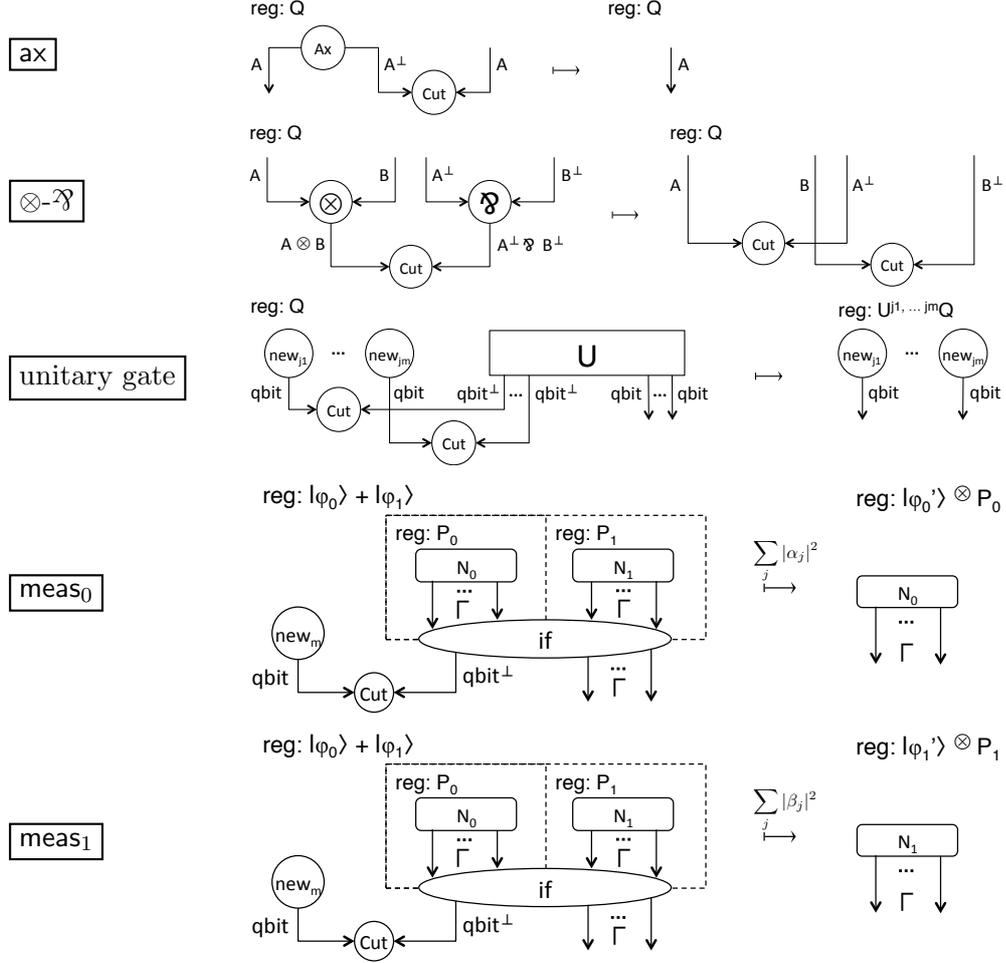
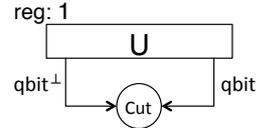


Figure 3.3: reduction rules of MLLqm proof structures

3.2.3 MLLqm Proof Nets and the Correctness Criterion

Our view of MLLqm proof structures is that they are “extended quantum circuits” that allow formalization of higher-order quantum computation.

As usual with proof structures, however, Def. 3.6 does not exclude proof structures that carries no computational contents—to put it technically, those which have cut nodes that cannot be eliminated. This is mainly due to vicious “feedback loops,” as seen in the proof structure on the right. We exclude such feedback loops by imposing a *correctness criterion* that is similar to Danos and Regnier’s “connected and acyclic” one [7]. Then *proof nets* are proof structures that comply with the correctness criterion.



In the current quantum setting the challenge is to devise a graph-theoretic correctness condition for unitary gate nodes. We follow the idea in [6].

Definition 3.9 (Correctness graphs with quantum nodes). Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be a proof structure. A *correctness graph* of \mathcal{N} is an undirected graph obtained by applying the following operations to \mathcal{S} .

- Ignore directions of all edges.
- For each \wp node, choose one of the two premises and disconnect the other.

- For each unitary gate node, choose an arbitrary bijective correspondence between the sets of qbit^\perp edges and qbit edges. Remove the node and connect each correspondent pair of edges.
- For each meas node, ignore its branches.
- The other nodes (ax , cut , \otimes , new) are not changed.

Here is an example. The correctness graphs for the proof structure in the Fig. 3.4 are shown as the four undirected graphs in the Fig. 3.5. There are two choices for the \mathfrak{A} node and two for the unitary gate node.

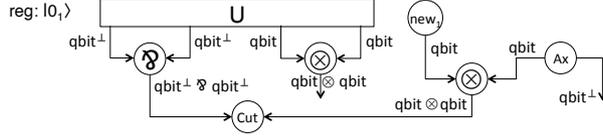


Figure 3.4: example of correctness criterion: proof net

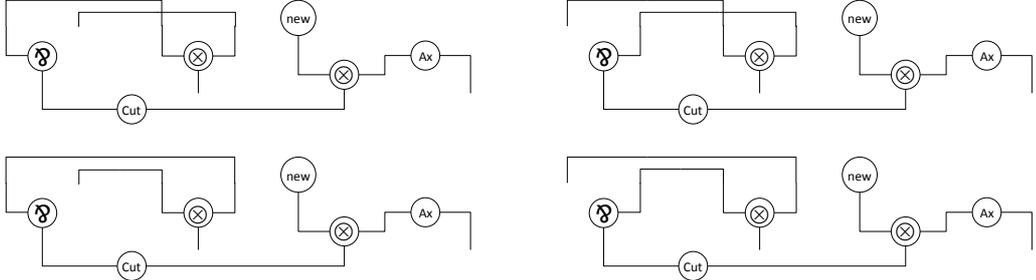


Figure 3.5: example of correctness criterion: correctness graphs

Definition 3.10 (MLLqm proof nets). A correctness graph is said to *satisfy the correctness criterion* if it is acyclic and connected.

A proof structure \mathcal{N} is called a *proof net* if each of its correctness graphs satisfies the correctness criterion and every branch in it is a proof net.

Lemma 3.11. *If a proof net \mathcal{N} reduces to another proof structure \mathcal{N}' (according to the rules in Def. 3.8), then \mathcal{N}' is also a proof net.*

Proof. By straightforward structural induction. See Appendix. \square

3.3 Translation of MLLqm Terms into Proof Nets

We assign a proof structure $\llbracket \Gamma \vdash M : A \rrbracket$ to each derivable type judgment $\Gamma \vdash M : A$. This turns out to satisfy the correctness criterion. Lem. 3.5 allows for the definition of $\llbracket \Gamma \vdash M : A \rrbracket$ by induction on derivation.

Definition 3.12 (Translation of terms into proof nets). For each derivable type judgment $\Gamma \vdash M : A$, a proof structure $\llbracket \Gamma \vdash M : A \rrbracket$ is inductively defined as in Fig. 3.6–3.7. For a type judgment $\Gamma \vdash M : A$, here we let $\llbracket \Gamma \vdash M : A \rrbracket = (\mathcal{S}_{\Gamma \vdash M : A}, \mathcal{Q}_{\Gamma \vdash M : A}, l_{\Gamma \vdash M : A})$, where Γ denotes a sequence A_1, A_2, \dots, A_n of types. In each case, the types A_j in the context Γ of $\Gamma \vdash M : A$ appear as their dual A_j^\perp in the conclusions of $\mathcal{S}_{\Gamma \vdash M : A}$.

The indexing l between new nodes and quantum registers are merged in the obvious way, in the cases of $\llbracket \Gamma \vdash \langle M, N \rangle : A \otimes B \rrbracket$ and $\llbracket \Gamma \vdash MN : B \rrbracket$. For example, suppose $l_1 : \{v_1, v_2\} \rightarrow \{1, 2\}$ with $v_1 \mapsto 1, v_2 \mapsto 2$, and $l_2 : \{v_3, v_4, v_5\} \rightarrow \{1, 2, 3\}$ with $v_3 \mapsto 2, v_4 \mapsto 1, v_5 \mapsto 3$. Then their merge $l' : \{v_1, v_2, v_3, v_4, v_5\} \rightarrow \{1, 2, 3, 4, 5\}$ can be defined by $v_1 \mapsto 1, v_2 \mapsto 2, v_3 \mapsto 4, v_4 \mapsto 3, v_5 \mapsto 5$.

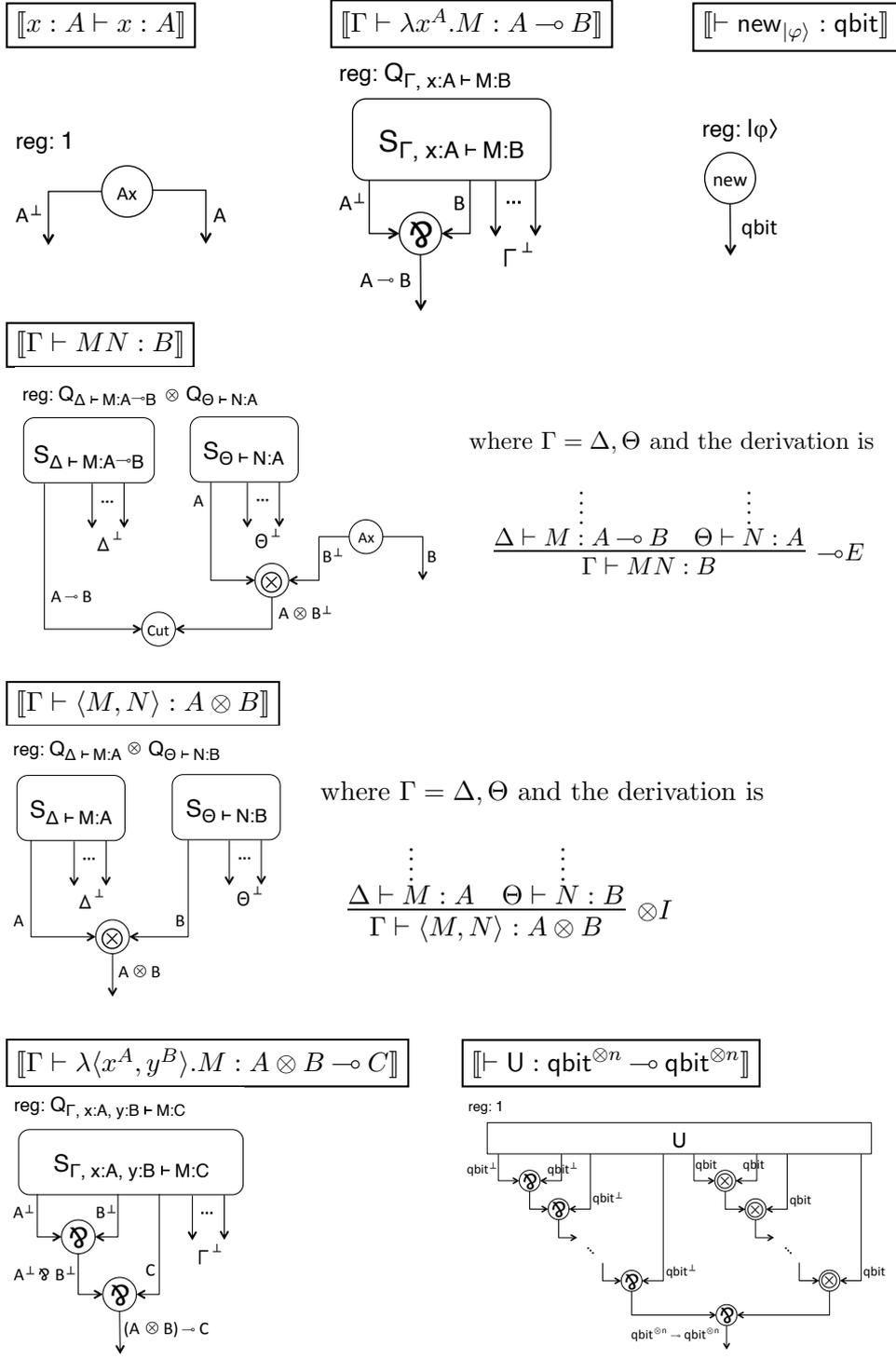


Figure 3.6: proof net translation of MLLqm terms—part I

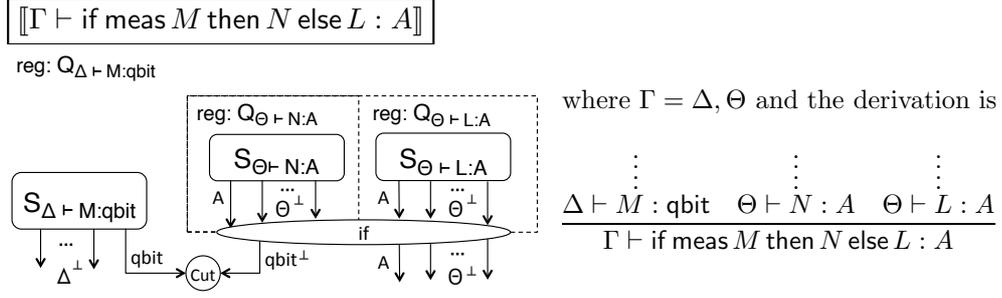


Figure 3.7: proof net translation of MLLqm terms—part II

Lemma 3.13. *For any derivable type judgment $\Gamma \vdash M : A$, the proof structure $\llbracket \Gamma \vdash M : A \rrbracket$ is a proof net.*

Proof. The fact that $\llbracket \Gamma \vdash M : A \rrbracket$ is a proof structure can be easily checked. It is proved to be a proof net by structural induction. For details see Appendix. \square

Hence, regarding MLLqm proof structures as a rewriting system for quantum computation, it is sufficient to consider solely MLLqm proof nets. This rewriting system exhibits the following pleasant properties (Thm. 3.14–3.15).

Theorem 3.14 (Termination of reduction). *The reduction of MLLqm proof nets is terminating.*

Proof. Reductions of unitary gate nodes and if nodes can occur at most equal to the number of them in a net. A reduction by \otimes - \wp rule strictly decrease the length of types on edges, a reduction by ax-cut rule strictly decrease the number of ax node, which are both finite. Therefore reduction can occur only finitely many times. \square

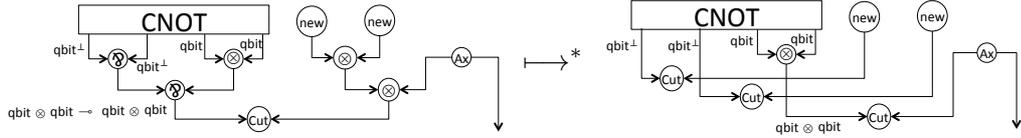
Regarding reduction of proof nets as cut elimination, it is natural to expect all the cut nodes to disappear after reduction terminates. This is unfortunately not the case and we have the following restricted result (Thm. 3.15). The condition in Thm. 3.15 corresponds to the condition that a term of MLLqm is *closed*, i.e. has no free variable. Intuitively, it states that a proof net “executes all computation steps” if the whole input is given. The condition also resembles that of *lazy cut elimination* for MALL proof nets [18].

Theorem 3.15 (Strong normalization). *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net. If no type containing qbit^{\perp} occurs in the conclusions of \mathcal{S} , then every maximal sequence of reductions from \mathcal{N} reaches a proof net that contains no cut nodes, no unitary gate nodes, or no if nodes.*

Proof. The proof is done by nested induction on the number of if nodes (including those inside branches) and the number of unitary gate nodes of level 0. For details see Appendix. \square

Remark 3.16. For MLL proof nets, one of the purposes to introduce correctness criteria in [7, 14] is to *characterize* those proof structures which arise from some proof in sequent calculus. Therefore the converse of Lem. 3.13—so-called *sequentialization*—is also proved in [14]. It allows (re)construction of sequent calculus proofs from proof nets.

However, sequentialization fails for MLLqm. Consider the following reduction; the original proof net is the translation of the term $\text{CNOT}(\text{new}_{|0\rangle}, \text{new}_{|0\rangle})$.



After two \otimes - \boxtimes reductions we do not yet get rid of the CNOT node; it is easily seen that there is no MLLqm term that gives rise to the resulting proof net.

This is a phenomenon that reflects the *nonlocal* character of MLLqm; and ultimately the nonlocality of quantum entanglement is to blame.

Sequentialization fails in general. Those proof nets which are sequentializable include: the net $\llbracket \Gamma \vdash M : A \rrbracket$ (trivially); and the normal form of the net $\llbracket \Gamma \vdash M : A \rrbracket$ for a closed term M . The latter is because Thm. 3.15 says that in that case the normal form is merely an MLL proof net with new nodes (associated with quantum register).

3.4 Examples and Discussion

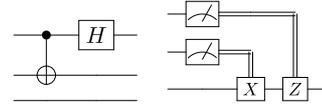
As syntax sugar we write $\langle x_1, x_2, x_3 \rangle \equiv \langle x_1, \langle x_2, x_3 \rangle \rangle$ and $\lambda \langle x_1^{A_1}, x_2^{A_2}, x_3^{A_3} \rangle. M \equiv \lambda \langle x_1^{A_1}, y^{A_2 \otimes A_3} \rangle. ((\lambda \langle x_2^{A_2}, x_3^{A_3} \rangle. M) y)$, where y is a fresh variable. Let

$$B := \lambda \langle x^{\text{qbit}}, y^{\text{qbit}}, z^{\text{qbit}} \rangle. \left((\lambda \langle v^{\text{qbit}}, w^{\text{qbit}} \rangle. (\text{H } v, w, z)) (\text{CNOT} \langle x, y \rangle) \right),$$

$$C := \lambda \langle s^{\text{qbit}}, t^{\text{qbit}}, u^{\text{qbit}} \rangle. (\text{if meas } s \text{ then } Z \text{ else } I) ((\text{if meas } t \text{ then } X \text{ else } I) u), \text{ and}$$

$$\beta_{00} := \text{CNOT} \langle \text{H new}_{|0\rangle}, \text{new}_{|0\rangle} \rangle$$

where H is the Hadamard gate, CNOT is the controlled not gate, I is the identity matrix, and Z and X are the Pauli matrices. The term β_{00} denotes one of the Bell state; and the terms B and C represent the quantum circuits on the right, respectively.



Quantum teleportation of one qubit $\alpha|0\rangle + \beta|1\rangle$ (where $\alpha, \beta \in \mathbb{C}$) is then described as an MLLqm term $T := (\lambda x^{\text{qbit}}. C(B \langle x, \beta_{00} \rangle)) \text{new}_{\alpha|0\rangle + \beta|1\rangle}$.

The term T is closed and has the type qbit . Its translation into proof net $\llbracket \vdash T : \text{qbit} \rrbracket$, after some reductions that are irrelevant to the quantum part, is shown in Fig. 3.8.

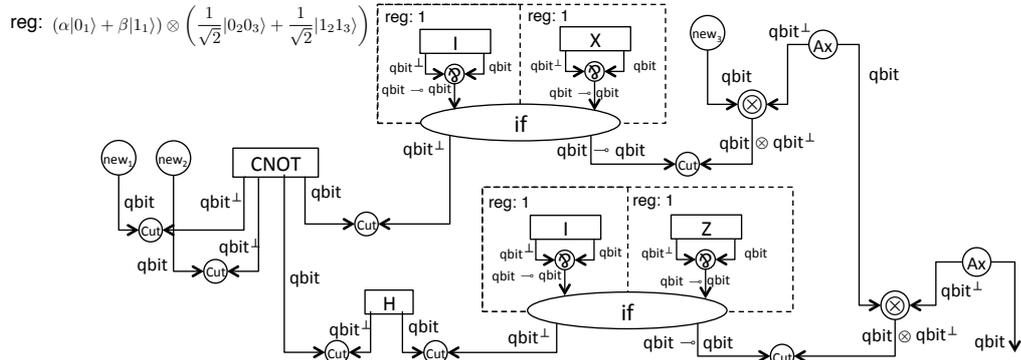


Figure 3.8: quantum teleportation (after some reductions irrelevant to the quantum part)

It is not hard to notice the similarity between the proof net in Fig. 3.8 and the presentation by a quantum circuit. In general, when we translate a first-order MLLqm term the resulting proof net looks quite much like a quantum circuit. Notice that the term T is indeed first-order.

Chapter 4

Token Machine Semantics for MLLqm Proof Nets

In this section we go on to introduce token machine semantics for MLLqm proof nets and prove its soundness, that is, the semantics is invariant under reduction of proof nets.

Token machines are one presentation of Girard's *geometry of interaction* [16]. Unlike the original presentation by C^* -algebras, token machines as devised in [27] are (concrete) automata and carry a strong operational flavor. See §2.1.3 for the definition of token machine semantics for MLL and [34] for more details.

The MLLqm token machines are different from the usual MLL ones in that it employs multiple tokens. Intuitively one token corresponds to one qubit; and they are required to synchronize when they go beyond a unitary gate node. This is one way how quantum entanglement (hence nonlocality) can be taken care of in token machine semantics. Use of multiple tokens is already in [6] where the style is called *wave-style token machine*. Multiple tokens inevitably results in *nondeterminism* in small-step behaviors of machines (which token moves first?). We prove confluence of small-step behaviors, and also uniqueness of big-step behaviors as its consequence. This is again like in [6].

In the current work we go beyond [6] and interpret measurements too. For that purpose we rely on the ideas developed in linear logic towards accommodating additive connectives: namely (*additive*) *slicing* of proof nets, and *weights* in token machines. See e.g. [18, 26].

4.1 Tokens

We start with usual definitions. We follow [26] most closely. The presentation in [34] is essentially the same. The definition of *context* is precisely the usual one:

Definition 4.1 (Context). A *context* is defined by the following BNF:

$$C ::= [] \mid C \otimes A \mid A \otimes C \mid C \wp A \mid A \wp C ,$$

where A is a type of MLLqm. Note that every context has exactly one *hole* $[]$. The type obtained by substituting a type A for the hole in a context C is denoted by $C[A]$. A context C is called a *context for* A if the type A is obtained by substituting some type B for the hole $[]$, i.e. $A \equiv C[B]$. The linear negation C^\perp of a context C is defined in a natural way, e.g. $(\text{qbit} \otimes [])^\perp := \text{qbit}^\perp \wp []$.

Definition 4.2 (Token). Given a proof net $\mathcal{N} = (\mathcal{S}, Q, l)$, a *token* is a 4-tuple (A, C, D, ζ) where

- A is an edge of \mathcal{S} (we abuse notations and identify an edge and the type occurrence A assigned to it; no confusion is likely),
- C is a context for A ,
- D is a *direction*, that is an element of $\{\uparrow, \downarrow\}$, and
- $\zeta \in \mathbb{N}$.

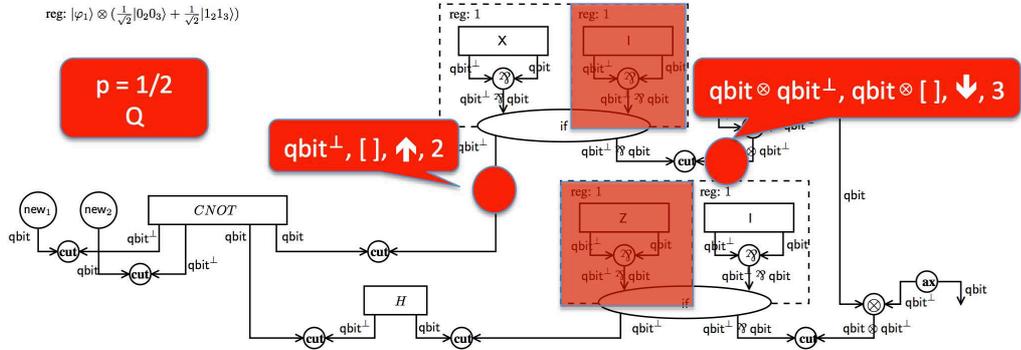
Note that A is identified with an *occurrence* rather than a *type* because the same type can have multiple occurrences in \mathcal{S} .

Intuitively, a token is a particle moving around the given proof net. The type occurrence A of a token indicates on which edge the token is. The context C designates which base type in A the token is concerned about. An example is $A \equiv \text{qbit}^\perp \wp \text{qbit}$ and $C \equiv [] \wp \text{qbit}$; token machine semantics is defined in such a way that a token's context determines which edge to take when the token hits a fork, namely a \wp node in the current example. The direction D of a token specifies whether it is going up or down along the edge.

Finally, the natural number ζ is a feature that is not in usual MLL proof nets: it records to which qubit of a quantum register the token corresponds. When a token is deployed the initial value of ζ is 0, meaning that the token does not yet know which qubit it corresponds to. When it hits a *new* node new_j , its index j is recorded in ζ .

4.2 The Token Machine $\mathcal{T}_{\mathcal{N}}$

Our goal is to construct a transition system (called a *token machine*) $\mathcal{T}_{\mathcal{N}}$ for a given MLLqm proof net \mathcal{N} . As an example, one state of the token machine is depicted below.



A state of $\mathcal{T}_{\mathcal{N}}$ is roughly the data that specifies the tokens in the proof net \mathcal{N} (how many of them, their locations, their contexts, etc.).

In the current setting of MLLqm a state carries much more data, in fact. For example it has a slicing, which is depicted by hatching the unselected branches in the above figure. It may feel strange that the selection of branches are specified even before the relevant qubits are measured: a *probability*—that is also carried by a state of a token machine ($p = 1/2$ in the above figure)—represents the likelihood of the slicing actually taken. The formal definition is as follows.

Definition 4.3 (State). Given a proof net $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$, a *state* of the token machine $\mathcal{T}_{\mathcal{N}}$ is a 5-tuple $(Q, p, b, T_{\text{pr}}, T_{\text{ms}})$ where

- Q is a quantum register,

- p is a probability, i.e. a real number satisfying $0 \leq p \leq 1$,
- b is a slicing (see Def. 3.7),
- T_{pr} is a finite set of tokens (called *principal tokens*),
- T_{ms} is another finite set of tokens (called *measurement tokens*).

A quantum register Q of a state is related to $Q_{\mathcal{N}}$ (that of the proof net) but not necessarily the same—this will be clarified by definitions below of the transition relation and the initial states of $\mathcal{T}_{\mathcal{N}}$.

We go on to define the transition structure $\rightarrow_{\mathcal{N}}$ of $\mathcal{T}_{\mathcal{N}}$ (Def. 4.4). We note that transitions $\rightarrow_{\mathcal{N}}$ form a binary relation between states—without any labels or probabilities assigned to transitions. Hence $\mathcal{T}_{\mathcal{N}}$ is simply a Kripke frame. We shall refer to the transitions $\rightarrow_{\mathcal{N}}$ in $\mathcal{T}_{\mathcal{N}}$ also as the *small-step semantics* of $\mathcal{T}_{\mathcal{N}}$.

The rules in Def. 4.4 are fairly complicated so their intuitions are stated first. The rules mainly describe how token(s) “move around the net.” Almost every rule moves only one token. An exception is the U-Apply rule: it makes tokens “synchronized” and moves them at once. The if-Meas rule deletes one measurement token. The U-Apply and if-Meas rules also act on the quantum register and the probability of a state, reflecting the quantum effects of the corresponding operations. A slicing b is left untouched by transitions.

Definition 4.4 (Transition $\rightarrow_{\mathcal{N}}$ of the token machine $\mathcal{T}_{\mathcal{N}}$). The *transition relation* $\rightarrow_{\mathcal{N}}$ between states of the token machine $\mathcal{T}_{\mathcal{N}}$ is defined by the rules as in Fig. 4.1–4.2. Each rule except the U-Apply and if-Meas rules is divided into two rules, one for principal tokens and the other for measurement tokens.

For each rule, we informally depict the intended movement of token(s) too.

Hatching over a branch means the branch is not selected by the slicing.

Lemma 4.5 (One-step confluence). *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net. The transition relation $\rightarrow_{\mathcal{N}}$ of its token machine $\mathcal{T}_{\mathcal{N}}$ is one-step confluent. That is, if both $s \rightarrow_{\mathcal{N}} s_1$ and $s \rightarrow_{\mathcal{N}} s_2$ hold, then either $s_1 = s_2$ or there exists a state s' such that $s_1 \rightarrow_{\mathcal{N}} s'$ and $s_2 \rightarrow_{\mathcal{N}} s'$.*

Proof. The case of $s_1 = s_2$ is trivial. Otherwise, by inspecting transition rules, it can be checked that two transitions $s \rightarrow_{\mathcal{N}} s_1$ and $s \rightarrow_{\mathcal{N}} s_2$ are on different tokens, say t_1 and t_2 . Then we can also check that “first move t_1 then t_2 ” and “first move t_2 then t_1 ” yield the same state. For details see Appendix. \square

4.3 Big-Step Semantics of $\mathcal{T}_{\mathcal{N}}$

We identify the “computational content” of a proof net \mathcal{N} to be the *big-step semantics* of the token machine $\mathcal{T}_{\mathcal{N}}$ that is defined below. The big-step semantics is intuitively the correspondence between an *initial state* $s \in I_{\mathcal{N}}$ and a *final state* $s' \in F_{\mathcal{N}}$, such that s reaches s' via a succession of $\rightarrow_{\mathcal{N}}$. By confluence of $\rightarrow_{\mathcal{N}}$ (Lem. 4.5) such s' is shown to be unique if it exists (Prop. 4.12); hence the big-step semantics is given as a partial function $I_{\mathcal{N}} \rightarrow F_{\mathcal{N}}$. Later in §4.4 we will show *soundness*, that is, the big-step semantics is invariant under the reduction of proof nets (as defined in §3.2), modulo certain “quantum effects.”

We start with singling out some states of $\mathcal{T}_{\mathcal{N}}$ as *initial* and *final*.

Notation 4.6 (Q_b^v). Let $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$ be an MLLqm proof net, b be a slicing of \mathcal{N} , and v be an if node in \mathcal{S} . By Q_b^v we denote the quantum register associated with the branch designated by b .

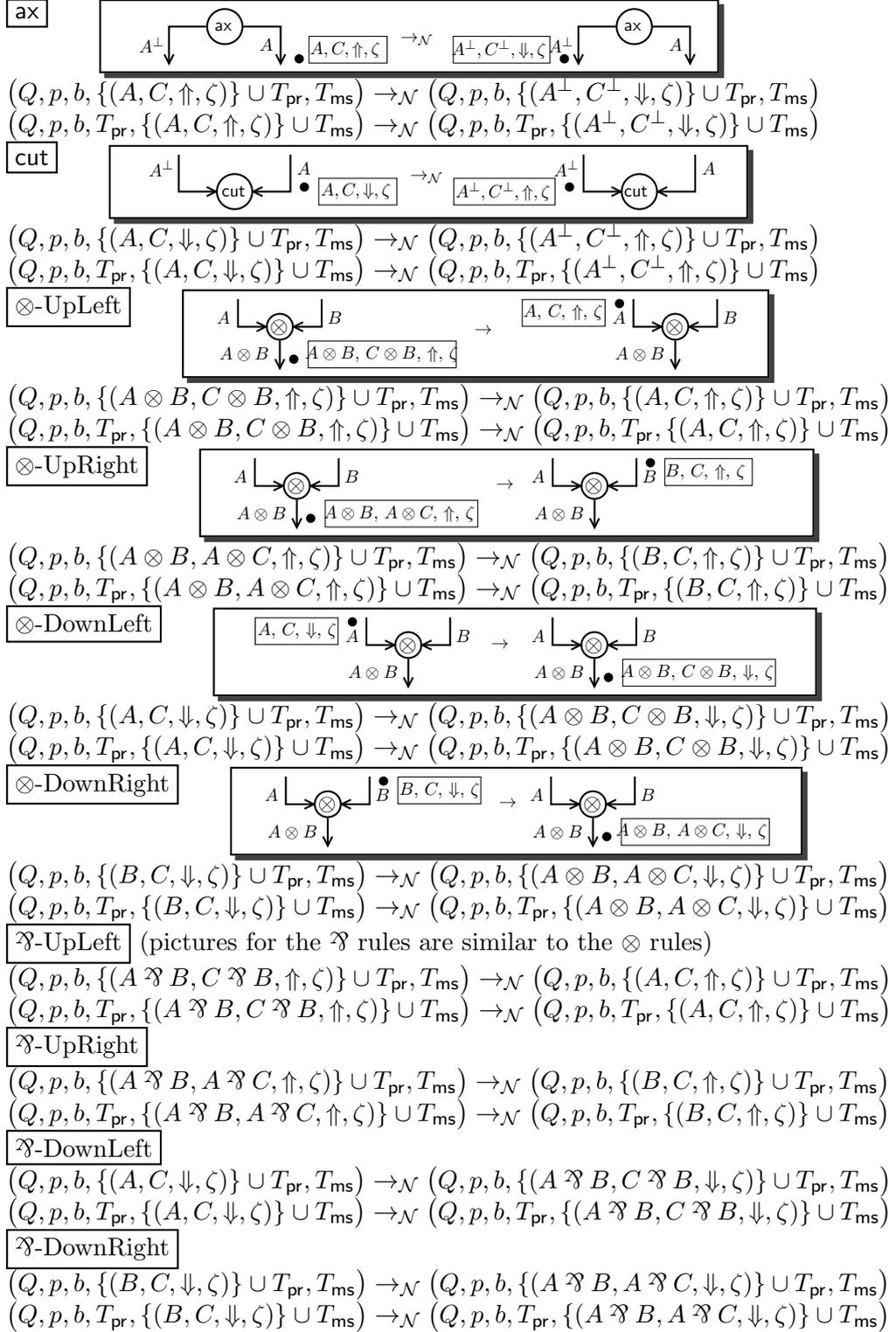


Figure 4.1: transition rules for $\mathcal{T}_{\mathcal{N}}$ —part I

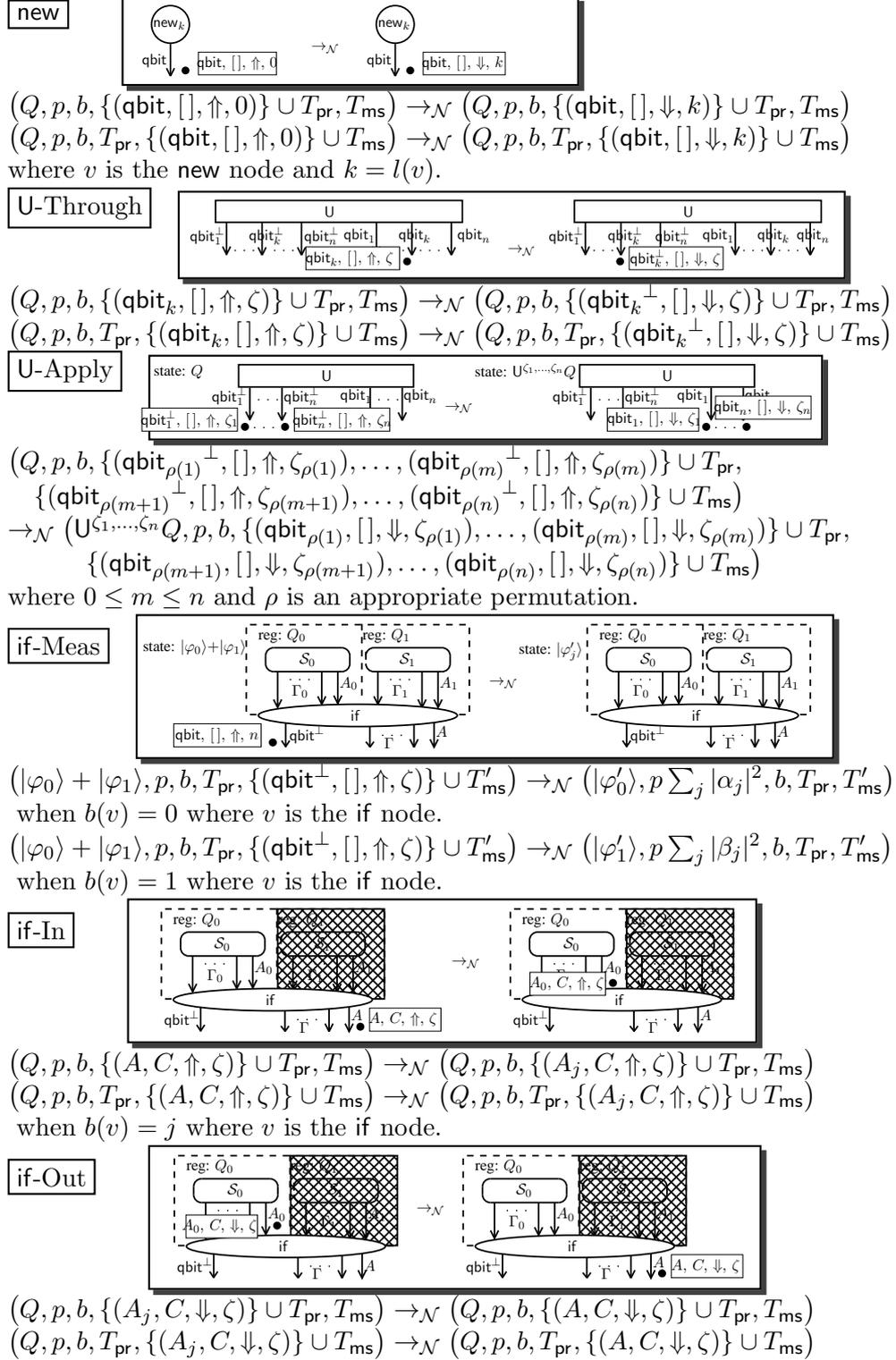


Figure 4.2: transition rules for $\mathcal{T}_{\mathcal{N}}$ —part II

Hence Q_b^v is a quantum register inside a dashed box attached to the if node v .

Definition 4.7 (Initial states). Let $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$ be an MLLqm proof net. A state $s = (Q, p, b, T_{\text{pr}}, T_{\text{ms}})$ of $\mathcal{T}_{\mathcal{N}}$ is said to be *initial* if:

- $Q = Q_{\mathcal{N}} \otimes \left(\bigotimes_{v \in V} Q_b^v \right)$ where V is the set of all if nodes in the slice $b(\mathcal{N})$ (of any level; recall Def. 3.7).
- A token (A, C, D, ζ) belongs to T_{pr} if and only if
 - A is a conclusion edge of level 0 (recall that we denote an edge by its type occurrence);
 - $C[\text{qbit}] \equiv A$; $D = \uparrow$; and $\zeta = 0$.
- A token (A, C, D, ζ) belongs to T_{ms} if and only if
 - $A \equiv \text{qbit}^\perp$, a query edge (one sticking left-down from an if node) in a branch remaining in the slice $b(\mathcal{N})$;
 - $C \equiv []$; $D = \downarrow$; and $\zeta = 0$.

The set of initial states is denoted by $I_{\mathcal{N}}$.

In an initial state, every principal token is at one of the conclusion edges (of level 0), waiting to go up. Measurement tokens are at query edges of any level (but only those which are in the slice $b(\mathcal{N})$). The quantum register Q keeps track not only of the level-0 register $Q_{\mathcal{N}}$ but also of “internal” registers (again which are in the slice $b(\mathcal{N})$).

Definition 4.8 (Final states). Let $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$ be an MLLqm proof net. A state $s = (Q, p, b, T_{\text{pr}}, T_{\text{ms}})$ of $\mathcal{T}_{\mathcal{N}}$ is said to be *final* if:

- each principal token $(A, C, D, \zeta) \in T_{\text{pr}}$ satisfies
 - A is a conclusion edge;
 - $C[\text{qbit}] = A$; and $D = \downarrow$.
- $T_{\text{ms}} = \emptyset$.

Therefore in a final state, all the principal tokens are back at conclusion edges, and all the measurement tokens are gone. Recall that the if-Meas transition in Def. 4.4 deletes a measurement token.

Definition 4.9 (Token Machine). The *token machine* for an MLLqm proof net \mathcal{N} is the 4-tuple $\mathcal{T}_{\mathcal{N}} = (S_{\mathcal{N}}, I_{\mathcal{N}}, F_{\mathcal{N}}, \rightarrow_{\mathcal{N}})$ where $S_{\mathcal{N}}$ is the set of states (Def. 4.3), $I_{\mathcal{N}}$ and $F_{\mathcal{N}}$ are the sets of initial and final states (Def. 4.7–4.8), and $\rightarrow_{\mathcal{N}} \subseteq S_{\mathcal{N}} \times S_{\mathcal{N}}$ is the (small-step) transition relation (Def. 4.4).

In what follows, the transitive closure of $\rightarrow_{\mathcal{N}}$ is denoted by $\rightarrow_{\mathcal{N}}^+$.

The following is the “semantics of proof nets by token machines.”

Definition 4.10 (Big-step semantics). Let \mathcal{N} be an MLLqm proof net. The *big-step semantics* of the token machine $\mathcal{T}_{\mathcal{N}}$, denoted by $\llbracket \mathcal{N} \rrbracket$, is the partial function

$$\llbracket \mathcal{N} \rrbracket : I_{\mathcal{N}} \rightarrow F_{\mathcal{N}} \text{ defined by } \llbracket \mathcal{N} \rrbracket(s) := \begin{cases} s' \in F_{\mathcal{N}} & \text{if } s \rightarrow_{\mathcal{N}}^+ s'; \\ \perp & \text{otherwise.} \end{cases}$$

Prop. 4.12 below exhibits the legitimacy of this definition (as a partial function). It is not *total* but *partial* in general: partiality arises when the conclusion contains a \mathbf{qbit}^\perp . For the proof nets translated from *closed* MLLqm terms, it is always total (Cor. 4.16).

Lemma 4.11 (Termination of transition). *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net. There is no infinite sequence of small-step transitions $\rightarrow_{\mathcal{N}}$ in $\mathcal{T}_{\mathcal{N}}$.*

Proof. Movement of a token can be uniquely traced back using the information carried by the token. Thus, given a token, the sequence of transitions on the token from initial state is unique.

Assume that there is an infinite sequence of transitions. It must contain two states that have the same token because of finiteness of the net and the number of tokens. Both states have the same sequence of transitions that starts from the position in an initial state. However, there is no transition that moves a token to its initial position, therefore such a sequence can occur only once. Contradiction. \square

Proposition 4.12 (Unique final state). *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net. If $s \rightarrow_{\mathcal{N}}^+ s_0$ and $s \rightarrow_{\mathcal{N}}^+ s_1$ with $s_0, s_1 \in F_{\mathcal{N}}$, then $s_0 = s_1$.*

Proof. By Lem. 4.5, $\rightarrow_{\mathcal{N}}$ is globally confluent. It is obvious too that final states are in normal form (i.e. no outgoing $\rightarrow_{\mathcal{N}}$). These two facts yield the claim. \square

4.4 Soundness of the Token Machine Semantics

Soundness of the big-step semantics—that it is invariant under the reduction of proof nets—holds only modulo certain quantum effects. The latter are formalized as follows, as suitable transformations of token machine states.

Definition 4.13 (\overline{U}). Let $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$ be an MLLqm proof net. Assume that there is a unitary gate node U in \mathcal{N} for which the unitary gate reduction rule in Fig. 3.3 can be applied, resulting in the proof net \mathcal{N}' . In this case, we define a function $\overline{U} : S_{\mathcal{N}} \rightarrow S_{\mathcal{N}'}$ by $\overline{U}(Q, p, b, T_{\text{pr}}, T_{\text{ms}}) := (U^{j_1, \dots, j_m} Q, p, b, T_{\text{pr}}, T_{\text{ms}})$.

Definition 4.14 ($\overline{\text{meas}}$). Let $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$ be an MLLqm proof net. Assume that there is an if node v in \mathcal{N} to which the meas_0 and meas_1 rules in Fig. 3.3 are applicable, resulting in nets \mathcal{N}_0 and \mathcal{N}_1 , respectively.

First we define functions $\overline{\text{meas}}_{|0\rangle}^v : I_{\mathcal{N}} \rightarrow I_{\mathcal{N}_0}$ and $\overline{\text{meas}}_{|1\rangle}^v : I_{\mathcal{N}} \rightarrow I_{\mathcal{N}_1}$, by

$$\begin{aligned} \overline{\text{meas}}_{|0\rangle}^v(|\varphi_0\rangle + |\varphi_1\rangle, p, b, T_{\text{pr}}, \{(qbit^\perp, [], \Downarrow, \zeta)\} \cup T_{\text{ms}}) &:= (|\varphi'_0\rangle, p \sum_j |\alpha_j|^2, b_0, T_{\text{pr}}, T_{\text{ms}}), \\ \overline{\text{meas}}_{|1\rangle}^v(|\varphi_0\rangle + |\varphi_1\rangle, p, b, T_{\text{pr}}, \{(qbit^\perp, [], \Downarrow, \zeta)\} \cup T_{\text{ms}}) &:= (|\varphi'_1\rangle, p \sum_j |\beta_j|^2, b_1, T_{\text{pr}}, T_{\text{ms}}), \end{aligned}$$

where b_j is defined by $b_j(u) := b(u)$ on every if node u in the proof net \mathcal{N}_j ($j \in \{0, 1\}$). Here the token $(qbit^\perp, [], \Downarrow, \zeta)$ in the definition is on the query edge of v , and $|\varphi_0\rangle, |\varphi'_0\rangle, |\varphi_1\rangle, |\varphi'_1\rangle$ are registers as in (1) in §3.2.2.

Finally we define a function $\overline{\text{meas}}^v : I_{\mathcal{N}} \rightarrow I_{\mathcal{N}_0} + I_{\mathcal{N}_1}$ (here $+$ denotes disjoint union) by

$$\overline{\text{meas}}^v(s) := \begin{cases} \overline{\text{meas}}_{|0\rangle}^v(s) & \text{if } b(v) = 0, \\ \overline{\text{meas}}_{|1\rangle}^v(s) & \text{if } b(v) = 1, \end{cases} \quad \text{where } s = (|\varphi\rangle, p, b, T_{\text{pr}}, T_{\text{ms}}).$$

Intuitively, the function $\overline{\text{meas}}^v$ “deletes” the if node v together with relevant entries in the slicing b . A quantum register and a probability are updated too, in an obvious manner.

Using these state transformations our main result is stated as follows.

Theorem 4.15 (Soundness). *Let $\mathcal{N} \mapsto \mathcal{N}'$ be a reduction of MLLqm proof nets. Then,*

1. $\llbracket \mathcal{N} \rrbracket = \llbracket \mathcal{N}' \rrbracket$ if the reduction is by the ax-cut or the \otimes - \wp rule.
2. $\llbracket \mathcal{N} \rrbracket = \llbracket \mathcal{N}' \rrbracket \circ \bar{\mathbf{U}}$ if the reduction is by the unitary gate rule, where \mathbf{U} is the corresponding unitary matrix.
3. $\llbracket \mathcal{N} \rrbracket \simeq (\llbracket \mathcal{N}_0 \rrbracket + \llbracket \mathcal{N}_1 \rrbracket) \circ \overline{\text{meas}}^v$ if the reduction is by one of the meas rules. In this case there must be another reduction possible due to the other meas rule, and we denote the resulting two proof nets by \mathcal{N}_0 and \mathcal{N}_1 (\mathcal{N}' is one of them). The function $\llbracket \mathcal{N}_0 \rrbracket + \llbracket \mathcal{N}_1 \rrbracket$ means case-distinction (recall the type $I_{\mathcal{N}} \rightarrow I_{\mathcal{N}_0} + I_{\mathcal{N}_1}$ of $\overline{\text{meas}}^v$). Here the equivalence \simeq is a natural identification of final states of $\mathcal{T}_{\mathcal{N}}, \mathcal{T}_{\mathcal{N}_0}$ and $\mathcal{T}_{\mathcal{N}_1}$. That is, $F \simeq G \stackrel{\text{def.}}{\iff} \forall x. F(x) \sim G(x)$ and $s \sim s' \stackrel{\text{def.}}{\iff} s = s'$ disregarding slicings.

Pictorially, the statements 2. and 3. say the following diagrams commute:

$$\begin{array}{ccc}
 I_{\mathcal{N}} \xrightarrow{\llbracket \mathcal{N} \rrbracket} F_{\mathcal{N}} & & I_{\mathcal{N}} \xrightarrow{\llbracket \mathcal{N} \rrbracket} F_{\mathcal{N}} \\
 \bar{\mathbf{U}} \downarrow & \parallel & \downarrow \sim \\
 I_{\mathcal{N}'} \xrightarrow{\llbracket \mathcal{N}' \rrbracket} F_{\mathcal{N}'} & & I_{\mathcal{N}_0} + I_{\mathcal{N}_1} \xrightarrow{\llbracket \mathcal{N}_0 \rrbracket + \llbracket \mathcal{N}_1 \rrbracket} F_{\mathcal{N}_0} + F_{\mathcal{N}_1} .
 \end{array}$$

Proof. Assume $\llbracket \mathcal{N} \rrbracket(s) = s'$. Then there exists a sequence of transitions $s \rightarrow_{\mathcal{N}} s_1 \rightarrow_{\mathcal{N}} s_2 \rightarrow_{\mathcal{N}} \dots \rightarrow_{\mathcal{N}} s'$ by definition of $\llbracket \mathcal{N} \rrbracket$. We can “permute” the sequence and obtain another sequence $s \rightarrow_{\mathcal{N}'}^* s'$ by which we can easily verify the statement. Conversely, if $\llbracket \mathcal{N}' \rrbracket(s) = s'$ we can build a sequence $s \rightarrow_{\mathcal{N}} s'$ from $s \rightarrow_{\mathcal{N}'} s'$. See Appendix for details. \square

Thm. 4.15 together with Thm. 3.15 yield the following corollary (Cor. 4.16). This corollary implies that the computation of a closed term ends *with a result*.

Corollary 4.16. *Let \mathcal{N} be a proof net with no qbit^\perp in its conclusions. Then the big-step semantics $\llbracket \mathcal{N} \rrbracket$ is total.*

Proof. Let \mathcal{N} be a proof net with no qbit^\perp in its conclusions. Then by Thm. 3.15 \mathcal{N} eventually reduces to a proof net \mathcal{N}' that contains no cut nodes, no unitary gate nodes, or no if nodes. Applying Thm. 4.15, $\llbracket \mathcal{N} \rrbracket$ is equal to a composition of $\llbracket \mathcal{N}' \rrbracket$, $\bar{\mathbf{U}}$'s, and $\overline{\text{meas}}^v$'s. It is easy to see that $\llbracket \mathcal{N}' \rrbracket$ is total and the functions $\bar{\mathbf{U}}$ and $\overline{\text{meas}}^v$ are total for any \mathbf{U} and v . Hence $\llbracket \mathcal{N} \rrbracket$ is also total. Note that difference of slices (that is identified by \simeq) is not relevant to totality. \square

4.5 Example

As a concrete example we briefly look at the token machine for the proof net for quantum teleportation (Fig. 3.8); we shall demonstrate that the qubit $\alpha|0_1\rangle + \beta|1_1\rangle$ (“stored” in the node new_1) is transmitted correctly.

The initial states of our interests are the following four:

$$(Q, 1, b_{ij}, \{(\text{qbit}, [], \uparrow, 0)\}, \{(\text{qbit}_x^\perp, [], \downarrow, 0), (\text{qbit}_z^\perp, [], \downarrow, 0)\}) ,$$

where Q is the quantum register $(\alpha|0_1\rangle + \beta|1_1\rangle) \otimes \left(\frac{1}{\sqrt{2}}|0_20_3\rangle + \frac{1}{\sqrt{2}}|1_21_3\rangle\right)$ and $i, j \in \{0, 1\}$. Each initial state (with a different slicing b_{ij}) corresponds to possible outcomes of the two measurements. Note that each has the probability 1.

It is straightforward to see that each of the four initial states is led to the final state $(\alpha|0\rangle + \beta|1\rangle, 1/4, b_{ij}, \{\text{qbit}, [], \downarrow, \mathbf{3}\}, \emptyset)$, with the qubit $\alpha|0\rangle + \beta|1\rangle$ assigned to the node `new3`. The probabilities (1/4 each) add up to 1 with the four initial states together, a fact which witnesses that the original qubit is successfully transmitted with the probability 1.

Chapter 5

Conclusions and Future Work

We introduced a quantum linear lambda calculus MLLq_m and a notion of MLLq_m proof net, into which terms of MLLq_m are translated. It is the first one that accommodates measurements as proof structures, and has suitable features for expressing higher-order computation thus going beyond quantum circuits. The definition of MLLq_m proof net is done by a correctness criterion that is similar to Danos and Regnier’s well-known one, which guarantees several properties of MLLq_m proof nets in the usual way of argument in linear logic. Therefore we consider our proof nets as an appropriate extension of MLL proof nets rather than an eccentric modification.

The GoI semantics (in the style of token machines) with measurements in this thesis is also the first one, which was mentioned in [6] as one of future work. The ideas of using a form of “weakening” to capture measurements (qubits are deleted) and that states of a token machine carry probabilities are new and clean, while the overall structure of the machine follows again the usual notion of slices used in linear logic.

We have not put much emphasis on the idea of synchronization at unitary gate nodes, but it also deserves further examination. Indeed, we have developed another notion of “ MLL proof net with synchronization” [5]; there we separate the layer of synchronization and the layer of (quantum) data, which makes the structure of computation clearer.

As future work, a long-term goal we set is to develop a compiler for a quantum programming language, with semantics-based correctness guarantee. Since GoI has been proved to be useful in construction of compilers for classical computation through its concreteness and game-theoretic nature (e.g. [10, 13], [27]), we believe that it would also be a valuable technique in compilation of quantum programming languages.

The current limitation of this work is that the expressiveness of the calculus is not enough, in the sense that it does not allow duplicable data or recursion. To achieve the long-term goal mentioned above, we aim at lifting this limitation one by one. Thus one direction is to accommodate duplicable data, namely the bit type. Although linear logic has a standard tool—the $!$ modality—to handle such data, there are subtle problems coming from the no-cloning property, nonlocality, etc. Another is to accommodate recursion. We expect to be able to adapt the techniques developed in [27] and [25]. With recursion, the language gets much expressive and it would be a nice option to write down with quantum procedures.

Simplifying and revising the token machine could also be possible: despite it behaves well, in the current definition it has a little too much information. Moreover, totality of the big-step semantics on non-closed terms could also be obtained by some modification.

References

- [1] 19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings. IEEE Computer Society (2004)
- [2] Abramsky, S., Coecke, B.: A categorical semantics of quantum protocols. In: LICS [1], pp. 415–425
- [3] Ball, T., Sagiv, M. (eds.): Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011. ACM (2011)
- [4] Dal Lago, U., Faggian, C.: On multiplicative linear logic, modality and quantum circuits. In: Jacobs, B., Selinger, P., Spitters, B. (eds.) QPL. EPTCS, vol. 95, pp. 55–66 (2011)
- [5] Dal Lago, U., Faggian, C., Hasuo, I., Yoshimizu, A.: Geometry of synchronization, submitted
- [6] Dal Lago, U., Zorzi, M.: Wave-style token machines and quantum lambda calculi (2013)
- [7] Danos, V., Regnier, L.: The structure of multiplicatives. Arch. for Math. Logic 28(3), 181–203 (1989)
- [8] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
- [9] de Falco, L.T.: Obsessional experiments for linear logic proof-nets. Mathematical Structures in Computer Science 13(6), 799–855 (2003)
- [10] Ghica, D.R.: Geometry of synthesis: a structured approach to vlsi design. In: Hofmann, M., Felleisen, M. (eds.) POPL. pp. 363–375. ACM (2007)
- [11] Ghica, D.R., Smith, A.: Geometry of synthesis ii: From games to delay-insensitive circuits. Electr. Notes Theor. Comput. Sci. 265, 301–324 (2010)
- [12] Ghica, D.R., Smith, A.: Geometry of synthesis iii: resource management through type inference. In: Ball and Sagiv [3], pp. 345–356
- [13] Ghica, D.R., Smith, A., Singh, S.: Geometry of synthesis iv: compiling affine recursion into static hardware. In: Chakravarty, M.M.T., Hu, Z., Danvy, O. (eds.) ICFP. pp. 221–233. ACM (2011)
- [14] Girard, J.Y.: Linear logic. Theor. Comput. Sci. 50, 1–102 (1987)
- [15] Girard, J.Y.: Geometry of interaction 2: deadlock-free algorithms. In: Martin-Löf, P., Mints, G. (eds.) Conference on Computer Logic. Lecture Notes in Computer Science, vol. 417, pp. 76–93. Springer (1988)

- [16] Girard, J.Y.: Geometry of interaction 1: Interpretation of system F. *Logic Colloquium* 88 (1989)
- [17] Girard, J.Y.: Geometry of interaction iii: Accommodating the additives. In: *Advances in Linear Logic*, LNS 222, CUP, 329-389. pp. 329–389. Cambridge University Press (1995)
- [18] Girard, J.Y.: Proof-nets: The parallel syntax for proof-theory. In: *Logic and Algebra*. pp. 97–124. Marcel Dekker (1996)
- [19] Gisin, N., Methot, A.A., Scarani, V.: Pseudo-telepathy: input cardinality and Bell-type inequalities. *International Journal of Quantum Information* 5(04), 525–534 (2007)
- [20] Gonthier, G., Abadi, M., Lévy, J.J.: The geometry of optimal lambda reduction. In: Sethi, R. (ed.) *POPL*. pp. 15–26. ACM Press (1992)
- [21] Green, A.S., Lumsdaine, P.L., Ross, N.J., Selinger, P., Valiron, B.: Quipper: a scalable quantum programming language. In: Boehm, H.J., Flanagan, C. (eds.) *PLDI*. pp. 333–342. ACM (2013)
- [22] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) *STOC*. pp. 212–219. ACM (1996)
- [23] Hasuo, I., Hoshino, N.: Semantics of higher-order quantum computation via geometry of interaction. In: *LICS*. pp. 237–246. IEEE Computer Society (2011)
- [24] Heinosaari, T., Ziomani, M.: *The Mathematical Language of Quantum Theory: From Uncertainty to Entanglement*. Cambridge University Press (2012), <http://books.google.co.jp/books?id=cZ8vd1nTIOEC>
- [25] Hoshino, N.: A modified GoI interpretation for a linear functional programming language and its adequacy. In: Hofmann, M. (ed.) *FOSSACS. Lecture Notes in Computer Science*, vol. 6604, pp. 320–334. Springer (2011)
- [26] Laurent, O.: A token machine for full geometry of interaction. In: *TLCA*. pp. 283–297 (2001)
- [27] Mackie, I.: The geometry of interaction machine. In: *POPL*. pp. 198–208 (1995)
- [28] Mairson, H.G., Terui, K.: On the computational complexity of cut-elimination in linear logic. In: Blundo, C., Laneve, C. (eds.) *ICTCS. Lecture Notes in Computer Science*, vol. 2841, pp. 23–36. Springer (2003)
- [29] Malherbe, O., Scott, P., Selinger, P.: Presheaf models of quantum computation: An outline. In: Coecke, B., Ong, L., Panangaden, P. (eds.) *Computation, Logic, Games, and Quantum Foundations. Lecture Notes in Computer Science*, vol. 7860, pp. 178–194. Springer (2013)
- [30] Melliès, P.A.: *Categorical semantics of linear logic, Panoramas et Synthèses*, vol. 27, chap. 1, pp. 15–215. Société Mathématique de France (2009)
- [31] Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge Univ. Press (2000)

- [32] Ömer, B.: Quantum programming in QCL. Master’s thesis, Institute of Information Systems, Technical University of Vienna (2000)
- [33] Petersen, L., Harper, R., Crary, K., Pfenning, F.: A type theory for memory allocation and data layout. In: Aiken, A., Morrisett, G. (eds.) POPL. pp. 172–184. ACM (2003)
- [34] Pinto, J.S.: Implantation Parallèle avec la Logique Linéaire (Applications des Réseaux d’Interaction et de la Géométrie de l’Interaction). Ph.D. thesis, École Polytechnique (2001), Main text in English
- [35] Selinger, P., Valiron, B.: Quantum lambda calculus. In: Gay, S., Mackie, I. (eds.) Semantic Techniques in Quantum Computation, pp. 135–172. Cambridge Univ. Press (2009)
- [36] Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: FOCS. pp. 124–134. IEEE Computer Society (1994)
- [37] Terui, K.: Proof nets and boolean circuits. In: LICS [1], pp. 182–191
- [38] van Tonder, A.: A lambda calculus for quantum computation. SIAM J. Comput. 33(5), 1109–1135 (2004)
- [39] Tov, J.A., Pucella, R.: Practical affine types. In: Ball and Sagiv [3], pp. 447–458
- [40] Yoshimizu, A., Hasuo, I., Faggian, C., Dal Lago, U.: Measurements in proof nets as higher-order quantum circuits. In: ESOP (2014), to appear

Appendix A

Appendix

A.1 Proof of Lem. 3.5

Note that the rule right above a judgment $\Gamma \vdash M : A$ in its derivation is uniquely determined by the structure of M . The notation $\text{FV}(M)$ denotes *free variables* of a term M .

Definition A.1 (Free Variables). The *free variables* of a term M of MLLqm, denoted by $\text{FV}(M)$, is inductively defined as follows:

- $\text{FV}(x) := \{x\}$
- $\text{FV}(\lambda x^A.M) := \text{FV}(M) \setminus \{x\}$
- $\text{FV}(\lambda \langle x^A, y^B \rangle.M) := \text{FV}(M) \setminus \{x, y\}$
- $\text{FV}(\langle M, N \rangle) := \text{FV}(M) \cup \text{FV}(N)$
- $\text{FV}(MN) := \text{FV}(M) \cup \text{FV}(N)$
- $\text{FV}(\text{new}_{|\varphi}) := \emptyset$ for all φ
- $\text{FV}(U) := \emptyset$ for all U
- $\text{FV}(\text{if meas } M \text{ then } N \text{ else } L) := \text{FV}(M) \cup \text{FV}(N) \cup \text{FV}(L)$

Lemma A.2. *If a type judgment $\Gamma \vdash M : A$ is derivable, then $|\Gamma| = \text{FV}(M)$.*

Proof. By structural induction on M .

- If $M \equiv x$, then the derivation must be $\frac{}{x : A \vdash x : A} \text{ax}$ and clearly $|\Gamma| = \text{FV}(M) = \{x\}$.
- If $M \equiv \text{new}_{|\varphi}$ or $M \equiv U$, then the derivation is also unique and has empty context. Thus $|\Gamma| = \text{FV}(M) = \emptyset$.
- If $M \equiv \lambda x^B.N$ and $A \equiv B \multimap C$, then a derivation is in the form

$$\frac{\begin{array}{c} \vdots \\ \Gamma, x : B \vdash N : C \end{array}}{\Gamma \vdash \lambda x^B.N : B \multimap C} \multimap I_1.$$

By the induction hypothesis $|\Gamma, x : B| = |\Gamma| \cup \{x\} = \text{FV}(N)$. Therefore $|\Gamma| = (|\Gamma| \cup \{x\}) \setminus \{x\} = \text{FV}(N) \setminus \{x\} = \text{FV}(\lambda x^B.N)$.

- If $M \equiv \lambda \langle x^B, y^C \rangle.N$ and $A \equiv B \otimes C \multimap D$, the proof is similar.

- If $M \equiv \langle N, L \rangle$ and $A \equiv B \otimes C$, then a derivation is

$$\frac{\frac{\vdots}{\Delta \vdash N : B} \quad \frac{\vdots}{\Theta \vdash L : C}}{\Gamma \vdash \langle N, L \rangle : B \otimes C} \otimes I$$

for some contexts Δ and Θ . By the induction hypothesis $|\Delta| = \text{FV}(N)$ and $|\Theta| = \text{FV}(L)$. Thus $|\Gamma| = |\Delta, \Theta| = |\Delta| \cup |\Theta| = \text{FV}(N) \cup \text{FV}(L) = \text{FV}(\langle N, L \rangle)$.

- If $M \equiv NL$, then a derivation is

$$\frac{\frac{\vdots}{\Delta \vdash N : B \multimap A} \quad \frac{\vdots}{\Theta \vdash L : B}}{\Gamma \vdash NL : A} \multimap E$$

for some type B and some contexts Δ and Θ . By the induction hypothesis $|\Delta| = \text{FV}(N)$ and $|\Theta| = \text{FV}(L)$. Hence $|\Gamma| = |\Delta, \Theta| = |\Delta| \cup |\Theta| = \text{FV}(N) \cup \text{FV}(L) = \text{FV}(NL)$.

- If $M \equiv \text{if meas } N \text{ then } L \text{ else } K$, a derivation is

$$\frac{\frac{\vdots}{\Delta \vdash N : \text{qbit}} \quad \frac{\vdots}{\Theta \vdash L : A} \quad \frac{\vdots}{\Theta \vdash K : A}}{\Gamma \vdash \text{if meas } N \text{ then } L \text{ else } K : A} \text{ meas}$$

for some contexts Δ and Θ . By the induction hypothesis $|\Delta| = \text{FV}(N)$ and $|\Theta| = \text{FV}(L) = \text{FV}(K)$. Hence $|\Gamma| = |\Delta, \Theta| = |\Delta| \cup |\Theta| = |\Delta| \cup |\Theta| \cup |\Theta| = \text{FV}(N) \cup \text{FV}(L) \cup \text{FV}(K) = \text{FV}(\text{if meas } N \text{ then } L \text{ else } K)$. \square

Lemma A.3. *If two type judgments $\Gamma \vdash M : A$ and $\Gamma \vdash M : A'$ are both derivable, then $A \equiv A'$.*

Proof. By structural induction on M .

- If $M \equiv x$, then the derivation must be

$$\frac{}{x : A \vdash x : A} \text{ ax} \quad \text{and} \quad \frac{}{x : A' \vdash x : A'} \text{ ax}.$$

Since $\Gamma = x : A = x : A'$, we have $A \equiv A'$.

- If $M \equiv \text{new}_{|\varphi\rangle}$ or $M \equiv \text{U}$, then M 's type is clearly unique. Thus $A \equiv A'$.
- If $M \equiv \lambda x^B.N$ with $A \equiv B \multimap C$ and $A' \equiv B \multimap C'$, then derivations for the judgments are in the form

$$\frac{\frac{\vdots}{\Gamma, x : B \vdash N : C}}{\Gamma \vdash \lambda x^B.N : B \multimap C} \multimap I_1 \quad \text{and} \quad \frac{\frac{\vdots}{\Gamma, x : B \vdash N : C'}}{\Gamma \vdash \lambda x^B.N : B \multimap C'} \multimap I_1.$$

By the induction hypothesis $C \equiv C'$ and thus $B \multimap C \equiv B \multimap C'$.

- If $M \equiv \lambda \langle x^B, y^C \rangle.N$, the proof is similar.
- If $M \equiv \langle N, L \rangle$ with $A \equiv B \otimes C$ and $A' \equiv B' \otimes C'$, then derivations for the judgments are in the form

$$\frac{\frac{\vdots}{\Delta \vdash N : B} \quad \frac{\vdots}{\Theta \vdash L : C}}{\Gamma \vdash \langle N, L \rangle : B \otimes C} \otimes I \quad \text{and} \quad \frac{\frac{\vdots}{\Delta \vdash N : B'} \quad \frac{\vdots}{\Theta \vdash L : C'}}{\Gamma \vdash \langle N, L \rangle : B' \otimes C'} \otimes I.$$

By Lemma A.2, the two contexts Δ and Θ are uniquely determined by dividing the context Γ according to $\text{FV}(N)$ and $\text{FV}(L)$. By the induction hypothesis $B \equiv B'$ and $C \equiv C'$, hence $B \otimes C \equiv B' \otimes C'$.

- If $M \equiv NL$, derivations for the judgments are in the form

$$\frac{\Delta \vdash N : B \multimap A \quad \Theta \vdash L : B}{\Gamma \vdash NL : A} \multimap E \quad \text{and} \quad \frac{\Delta \vdash N : B' \multimap A' \quad \Theta \vdash L : B'}{\Gamma \vdash NL : A'} \multimap E.$$

By Lemma A.2, Δ and Θ are uniquely determined. By the induction hypothesis $B \equiv B'$ and $B \multimap A \equiv B' \multimap A'$. Therefore $A \equiv A'$.

- If $M \equiv \text{if meas } N \text{ then } L \text{ else } K$ derivations are in the form

$$\frac{\Delta \vdash N : \text{qbit} \quad \Theta \vdash L : A \quad \Theta \vdash K : A}{\Gamma \vdash \text{if meas } N \text{ then } L \text{ else } K : A} \text{meas}$$

and

$$\frac{\Delta \vdash N : \text{qbit} \quad \Theta \vdash L : A' \quad \Theta \vdash K : A'}{\Gamma \vdash \text{if meas } N \text{ then } L \text{ else } K : A'} \text{meas}.$$

By Lemma A.2, Δ and Θ is uniquely determined. Hence $A \equiv A'$ by the induction hypothesis. \square

Proof. (of Lem. 3.5) By structural induction on M .

- If $M \equiv x$, then the derivation must be $\frac{}{x : A \vdash x : A} \text{ax}$ and thus unique.
- Similarly, if $M \equiv \text{new}_{|\varphi\rangle}$ or $M \equiv \text{U}$ then the derivation is clearly unique.
- If $M \equiv \lambda x^B.N$ and $A \equiv B \multimap C$, then a derivation is in the form

$$\frac{\Gamma, x : B \vdash N : C}{\Gamma \vdash \lambda x^B.N : B \multimap C} \multimap I_1.$$

By the induction hypothesis the derivation of $\Gamma, x : B \vdash N : C$ is unique. Thus the derivation of $\Gamma \vdash \lambda x^B.N : B \multimap C$ is also unique.

- If $M \equiv \lambda \langle x^B, y^C \rangle.N$ and $A \equiv B \otimes C \multimap D$, the proof is similar.
- If $M \equiv \langle N, L \rangle$ and $A \equiv B \otimes C$, then a derivation is in the form

$$\frac{\Delta \vdash N : B \quad \Theta \vdash L : C}{\Gamma \vdash \langle N, L \rangle : B \otimes C} \otimes I.$$

By Lemma A.2 Δ and Θ is uniquely determined, thus the judgments $\Delta \vdash N : B$ and $\Theta \vdash L : C$ above the line are unique. Since the derivations of $\Delta \vdash M : B$ and $\Theta \vdash N : C$ are unique by the induction hypothesis, the derivation of $\Gamma \vdash \langle N, L \rangle : B \otimes C$ is also unique.

- If $M \equiv NL$, then by the typing rule $\multimap E$ a derivation is in the form

$$\frac{\Delta \vdash N : B \multimap A \quad \Theta \vdash L : B}{\Gamma \vdash NL : A} \multimap E.$$

The contexts Δ and Θ are uniquely determined by Lemma A.2 and the type B is unique by Lemma A.3. Thus the judgments $\Delta \vdash N : B \multimap A$ and $\Theta \vdash L : B$ are also unique. By the induction hypothesis the derivations of them are unique, therefore the derivation of $\Gamma \vdash NL : A$ is unique.

- If $M \equiv \text{if meas } N \text{ then } L \text{ else } K : A$ then a derivation is in the form

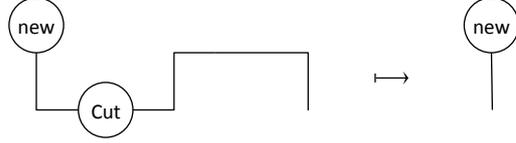
$$\frac{\begin{array}{c} \vdots \\ \Delta \vdash N : \text{qbit} \end{array} \quad \begin{array}{c} \vdots \\ \Theta \vdash L : A \end{array} \quad \begin{array}{c} \vdots \\ \Theta \vdash K : A \end{array}}{\Gamma \vdash \text{if meas } N \text{ then } L \text{ else } K : A} \text{ meas.}$$

The three judgments above the line are all unique and the derivations of them are also unique by the induction hypothesis. Hence the derivation of $\Gamma \vdash \text{if meas } N \text{ then } L \text{ else } K : A$ is unique. \square

A.2 Proof of Lem. 3.11

Proof. What we should show is about the correctness criterion, so quantum registers are not relevant here. The statement of this lemma is well-known to hold for MLL proof nets and the existence of quantum nodes does not affect it since the ax rule and the \otimes - \wp rule are purely local. For the remaining rules it is also easy:

- For the unitary gate rule, consider correctness graphs of \mathcal{N} and \mathcal{N}' . The reduction corresponds to the mapping



for each *new* node. It is clear that this does not break acyclicity and connectivity.

- For the *meas* rule 0 and 1, each branch is a proof net by definition. Replacing a *meas* node with a proof net with the same conclusion preserves correctness criterion. Thus if \mathcal{N} is a proof net, \mathcal{N}' is also a proof net. \square

A.3 Proof of Lem. 3.13

Proof. The fact that $\llbracket \Gamma \vdash M : A \rrbracket$ is a proof structure can be easily checked. It is proved to be a proof net by straightforward structural induction.

- The correctness graph of $\llbracket x : A \vdash x : A \rrbracket$ obviously satisfies the criterion.
- Connecting two conclusions of a proof structure by a \wp node does not yield any cycle nor disjoint components in its correctness graphs. Thus the correctness graphs of $\llbracket \Gamma \vdash \lambda x^A. M : A \multimap B \rrbracket$ satisfy the criterion if those of $\llbracket \Gamma, x : A \vdash M : B \rrbracket$ satisfy the criterion.
- Similarly the correctness graphs of $\llbracket \Gamma \vdash \lambda \langle x^A, y^B \rangle. M : A \otimes B \multimap C \rrbracket$ satisfy the criterion if those of $\llbracket \Gamma, x : A, y : B \vdash M : C \rrbracket$ do so.
- If all correctness graphs of $\llbracket \Delta \vdash M : A \multimap B \rrbracket$ and $\llbracket \Theta \vdash N : A \rrbracket$ satisfy the criterion, the construction of $\llbracket \Delta, \Theta \vdash MN : B \rrbracket$ makes the whole structure connected and does not yield any cycle in correctness graphs. Thus the correctness graphs of $\llbracket \Delta, \Theta \vdash MN : B \rrbracket$ satisfies the criterion.
- In the same way, the correctness graphs of $\llbracket \Delta, \Theta \vdash \langle M, N \rangle : A \otimes B \rrbracket$ satisfies the criterion if both $\llbracket \Delta \vdash M : A \rrbracket$ and $\llbracket \Theta \vdash N : B \rrbracket$ do so.
- The correctness graph of $\llbracket \vdash \text{new}_{|\varphi\rangle} : \text{qbit} \rrbracket$ clearly satisfies the criterion.

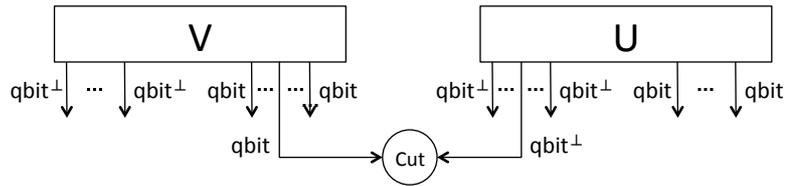
- The correctness graphs of $\llbracket \vdash U : \text{qbit}^{\otimes n} \multimap \text{qbit}^{\otimes n} \rrbracket$ also satisfy the criterion.
- The correctness graphs of $\llbracket \Delta, \Theta \vdash \text{if meas } M \text{ then } N \text{ else } L : A \rrbracket$ obviously satisfies the criterion if those of $\llbracket \Delta \vdash M : \text{qbit} \rrbracket$ do so. Both $\llbracket \Theta \vdash N : A \rrbracket$ and $\llbracket \Theta \vdash L : A \rrbracket$ are proof nets by the induction hypothesis. \square

A.4 Proof of Thm. 3.15

Proof. When a proof net \mathcal{N} has n if nodes (including those inside of branches) and m unitary gate nodes of level 0, we write $\#\mathcal{N} = (n, m)$. The proof is done by nested induction on them.

- i) $\#\mathcal{N} = (0, 0)$. Simply because of cut elimination property of MLL. The existence of new node is not relevant here: if a new node is connected to a cut node, the other edge of the cut node is necessarily connected to an ax node because of typing.
- ii) $\#\mathcal{N} = (0, m)$ with $0 < m$. We show that any sequence $\sigma : \mathcal{N} \mapsto^* \mathcal{N}'$ of reductions that has no unitary gate reduction is not maximal.

- If ax rule or \otimes - \wp rule can be applied to \mathcal{N}' then clearly σ is not maximal.
- Suppose both ax rule and \otimes - \wp rule cannot be applied to \mathcal{N}' . Then there are m unitary gate nodes in \mathcal{N}' since any one of them is not reduced while σ . Choose one of unitary gate nodes in \mathcal{N}' . If it can be reduced then σ is not maximal. Otherwise there must be some edges of type qbit^\perp that are not connected to new nodes via cut nodes. Choose one of those edges and start a traverse along the chosen edge and go down the structure. Because of the assumption that \mathcal{N} does not have any qbit^\perp in its conclusions, the traverse necessarily runs into a cut node. The cut cannot be between \otimes and \wp by the assumption that \otimes - \wp rule cannot be applied. Similarly it cannot be connected to an axiom node. Moreover it cannot be connected to new node nor if node. Hence the cut must immediately be connected to another unitary gate node, and considering types it turns out that the connection is in the form below.



If σ is maximal, any unitary gate node cannot be reduced and thus we must be able to continue such traverse infinitely. Since a proof net is finite, that implies we will revisit a unitary gate node while the traverse. However, if such revisiting can be done then at least one correctness graph of \mathcal{N}' has a cycle in it, which contradicts to the fact that \mathcal{N}' is a proof net. Therefore there exists at least one unitary gate node that can be reduced and σ is not maximal.

Hence by contraposition a maximal sequence of reductions must have a unitary gate reduction. Thus m will necessarily decrease.

- iii) $\#\mathcal{N} = (n, m)$ with $0 < n$ and $0 < m$. Similarly to the case ii), a maximal sequence of reductions necessarily contains a unitary gate reduction or a meas reduction. Thus either n or m will strictly decrease. \square

A.5 Proof of Lem. 4.5

We introduce the following notation. It will be useful in the proofs later.

Definition A.4. If a transition $s \rightarrow_{\mathcal{N}} s'$ is given by the rule r for the node v on the token t , we write $s \xrightarrow{a}_{\mathcal{N}} s'$ with $a := (r, v, t)$. We also write $s \xrightarrow{a_1 a_2 \dots a_m}_{\mathcal{N}} s'$ if $s \xrightarrow{a_1}_{\mathcal{N}} s_1 \xrightarrow{a_2}_{\mathcal{N}} s_2 \xrightarrow{a_3}_{\mathcal{N}} \dots \xrightarrow{a_m}_{\mathcal{N}} s'$ for some states s_1, s_2, \dots, s_{m-1} . Such a sequence $a_1 a_2 \dots a_m$ is denoted by σ, τ , etc.

Lemma A.5. *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be a proof net, and $s \xrightarrow{ab}_{\mathcal{N}} s'$. Assume further that b does not depend on a , that is, the token in b is not yielded by the transition $\xrightarrow{a}_{\mathcal{N}}$. Then we have $s \xrightarrow{ba}_{\mathcal{N}} s'$, too.*

Proof. Suppose $s \xrightarrow{ab}_{\mathcal{N}} s_1$ and $s \xrightarrow{ba}_{\mathcal{N}} s_2$. Observing transition rules, it can be checked that the slicings and the sets of tokens of s_1 and s_2 are the same. Thus it suffices to show that the quantum registers and the probabilities also coincide. If either a or b does not acts on quantum registers, it is clear. If both act on quantum registers, they act on disjoint qubits. Then the order of applying such two operations (unitary transformation or measurement) does not affect the resulting quantum register and probability since the tensor \otimes in vector space is monoidal: $(\mathcal{F} \otimes \text{id})(\text{id} \otimes \mathcal{E}) = \mathcal{F} \otimes \mathcal{E} = (\text{id} \otimes \mathcal{E})(\mathcal{F} \otimes \text{id})$ for any quantum operations \mathcal{F} and \mathcal{E} . Hence $s_1 = s_2$. \square

Proof. (as a corollary of Lem. A.5) Suppose $s \xrightarrow{a}_{\mathcal{N}} s_1$ and $s \xrightarrow{b}_{\mathcal{N}} s_2$. If $a = b$ then clearly $s_1 = s_2$. If $a \neq b$, then a and b are on different tokens (hence in particular b does not depend on a). In this case it is obvious that after $\xrightarrow{a}_{\mathcal{N}}$, a transition $\xrightarrow{b}_{\mathcal{N}}$ is possible. Let s' be such that $s \xrightarrow{ab}_{\mathcal{N}} s'$; then by Lem. A.5, we have $s \xrightarrow{ba}_{\mathcal{N}} s'$. \square

A.6 Proof of Thm. 4.15

The next lemma roughly says that, if an edge's type contains **qbit**, then a token with a context that designates that occurrence of **qbit** will eventually visit it.

Lemma A.6. *Let $\mathcal{N} = (\mathcal{S}, Q_{\mathcal{N}}, l)$ be an MLLqm proof net and $\llbracket \mathcal{N} \rrbracket(s) = s'$. Let A be an edge in \mathcal{S} of level 0 that $A \equiv C[\text{qbit}]$. Then any sequence of small-step transitions $s \rightarrow_{\mathcal{N}} s_1 \rightarrow_{\mathcal{N}} \dots \rightarrow_{\mathcal{N}} s'$ from s to s' contains a state $(Q, p, b, T_{\text{pr}}, T_{\text{ms}})$ in which a token (A, C, \uparrow, ζ) is in T_{pr} or T_{ms} .*

Proof. Assume such a token has just arrived the edge on which A occurs. Tracing back the transitions on that token yields a traverse along \mathcal{S} . If the traverse infinitely continues, it necessarily contain an infinite loop along \mathcal{S} . However, that situation implies there is a cycle that remains in at least one correctness graph. Hence the traverse eventually terminates and considering the type and context it ends in a conclusion or a query node, where a token start traveling in any initial state by definition. Hence indeed a token will arrive. \square

Corollary A.7. *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net that contains a unitary gate node U that can be reduced by the unitary gate rule in Fig. 3.3. Assume $\llbracket \mathcal{N} \rrbracket(s) = s'$. Then any sequence of small transitions $s \rightarrow_{\mathcal{N}} s_1 \rightarrow_{\mathcal{N}} \dots \rightarrow_{\mathcal{N}} s'$ from s to s' contains a transition by the U -Apply rule in Fig. 4.2.*

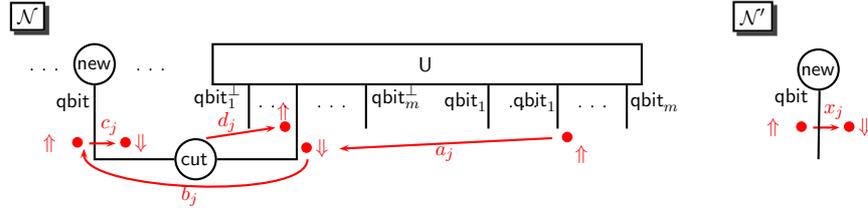
Proof. By Lem. A.6, for each new node that is connected to the node U and is to be reduced together, there must be a token that visit it. These tokens can only go beyond the node U by the U -Apply rule in Fig. 4.2; and they must do so to reach a final state s' . \square

The next lemma says that two different tokens cannot correspond to the same qubit.

Lemma A.8. *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net. Let s_1 be an initial state of $\mathcal{T}_{\mathcal{N}}$ and $s_1 \rightarrow_{\mathcal{N}} s_2 \rightarrow_{\mathcal{N}} s_3 \cdots \rightarrow_{\mathcal{N}} s_{m+1}$. For any state s_j ($j = 1, 2, \dots, m+1$) and any two distinct tokens (A_1, C_1, D_1, ζ_1) and (A_2, C_2, D_2, ζ_2) , if they occur in the same s_j then they satisfy $\zeta_1 \neq \zeta_2$ or $\zeta_1 = \zeta_2 = 0$.*

Proof. Since in an initial state all the tokens have $\zeta = 0$, the condition $\zeta_1 = \zeta_2 \neq 0$ implies that the two tokens have reached the same new node. However, tracing back transitions, it is easy to see that such tokens must have come from the same conclusion with the same context. This is prohibited by the definition of initial states. \square

Lemma A.9. *Let $\mathcal{N} = (\mathcal{S}, Q, l)$ be an MLLqm proof net with a unitary gate node U that can be reduced by the unitary gate rule in Fig. 3.3, resulting in the net \mathcal{N}' . Consider the following transitions in the token machines $\mathcal{T}_{\mathcal{N}}$ and $\mathcal{T}_{\mathcal{N}'}$.*



Here a_j is a U -Through transition; b_j and d_j are cut transitions; and c_j and x_j are new transitions. Then we have

$$s \xrightarrow{\tau a_1 b_1 c_1 d_1 a_2 b_2 c_2 d_2 \cdots a_m b_m c_m d_m u} \mathcal{N} s' \iff \bar{U}(s) \xrightarrow{\tau x_1 x_2 \cdots x_m} \mathcal{N}' s' ,$$

where s is an initial state, τ is an arbitrary sequence, and u is a U -Apply transition that involves the node U .

Proof. By induction on the number k of transitions contained in τ that affect quantum registers. Let Q be the quantum register of s . Let s_1 and s_2 be the states satisfying $s \xrightarrow{\tau a_1 b_1 c_1 d_1 a_2 b_2 c_2 d_2 \cdots a_m b_m c_m d_m u} \mathcal{N} s_1$ and $\bar{U}(s) \xrightarrow{\tau x_1 x_2 \cdots x_m} \mathcal{N}' s_2$. It can easily be checked that s_1 and s_2 are the same except their quantum registers. So it suffices to show that their quantum registers coincide.

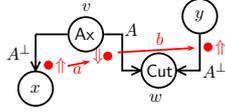
- Case $k = 0$. Since τ does not affect quantum registers, it is clear that the registers of s_1 and s_2 coincide with the register obtained by applying the unitary matrix U to Q . Thus $s_1 = s_2$.
- Case $k > 0$. Then τ can be written as $\tau_1 q_1 \tau_2 q_2 \cdots \tau_k q_k$ where q_j is a transition that acts on quantum register (i.e. by unitary gate rule or measurement rule), τ_j is a sequence of transitions that does not contain such quantum transitions. Considering U can be reduced, u and q_1 act on disjoint qubits, hence \bar{U} and q_1 act on disjoint qubits. Thus manipulating a quantum register first by \bar{U} and second by q_1 yields the same quantum register as first by q_1 and second by \bar{U} does. Hence $\bar{U}(s) \xrightarrow{\tau_1 q_1} \mathcal{N}' s_1 \Leftrightarrow s \xrightarrow{\tau_1 q_1} \mathcal{N}' s'_1$ with $\bar{U}(s'_1) = s_1$. Therefore

$$\begin{aligned}
& \bar{U}(s) \xrightarrow{\tau_1 q_1 \tau_2 q_2 \cdots \tau_k q_k x_1 x_2 \cdots x_m} \mathcal{N}' s' \\
& \Leftrightarrow s \xrightarrow{\tau_1 q_1} \mathcal{N}' s_1 \text{ and } \bar{U}(s_1) \xrightarrow{\tau_2 q_2 \cdots \tau_k q_k x_1 x_2 \cdots x_m} \mathcal{N}' s' \\
& \Leftrightarrow s \xrightarrow{\tau_1 q_1} \mathcal{N}' s_1 \xrightarrow{\tau_2 q_2 \cdots \tau_k q_k a_1 b_1 c_1 d_1 a_2 b_2 c_2 d_2 \cdots a_m b_m c_m d_m u} \mathcal{N}' s' \text{ (by induction hypothesis.)} \quad \square
\end{aligned}$$

Proof. (of Thm. 4.15) Proof sketch: for $\llbracket \mathcal{N} \rrbracket(s) = s'$, there exists a sequence of transitions $s \xrightarrow{\sigma} \mathcal{N}' s'$ by definition. Permuting σ by Lem. A.5, we obtain σ' in which all the relevant transitions adjoin and $s \xrightarrow{\sigma'} \mathcal{N}' s'$. For such a sequence σ' we can easily verify that $\llbracket \mathcal{N}' \rrbracket(s) = s'$ (or $(\llbracket \mathcal{N}' \rrbracket \circ \bar{U})(s) = s'$ or $(\llbracket \mathcal{N}' \rrbracket \circ \text{meas}_{|0\rangle}^v)(s) = s'$) also holds. Conversely, if $\llbracket \mathcal{N}' \rrbracket(s) = s'$ we can build a sequence $s \xrightarrow{\sigma'} \mathcal{N}' s'$ from $s \xrightarrow{\sigma'} \mathcal{N}' s'$.

Details:

1. We prove the case of ax-cut rule; the case of \otimes - \mathfrak{A} rule can be proved in similar way. Let v and w be the ax node and the cut node that are under reduction, x and y be the nodes that are connected to v and w respectively.



In case v has a pending edge, regard that it is connected to a dummy node with one incoming edge and no outgoing edge. Let $\llbracket \mathcal{N} \rrbracket(s) = s'$ for $s \in I_{\mathcal{N}}$. Then there is a transition relation $s \xrightarrow{\sigma} \mathcal{N}' s'$ by definition of $\llbracket \mathcal{N} \rrbracket$. Since we have only one ground type qbit, either A or A^\perp contains type qbit. Thus a token will necessarily come by Lem. A.6. Assume a token comes from x and goes to y . Then $s \xrightarrow{\sigma_1 a \sigma_2 b \sigma_3} \mathcal{N}' s'$ where a corresponds to the transition for v and b to the transition for w . By Lem. A.5, $s \xrightarrow{\sigma_1 a b \sigma_2 \sigma_3} \mathcal{N}' s'$ also holds since σ_2 cannot contain any transition on t . The ax-cut reduction does not change conclusions nor quantum registers, so the initial state s of \mathcal{N} is also an initial state of \mathcal{N}' and $s \xrightarrow{\sigma_1 \sigma_2 \sigma_3} \mathcal{N}' s'$ holds. Hence $\llbracket \mathcal{N}' \rrbracket(s) = s'$. The case the token comes from y and goes to x is similar.

Conversely, assume $\llbracket \mathcal{N}' \rrbracket(s) = s'$ for $s \in I_{\mathcal{N}'}$. If s and s' are related by a sequence of transition relations $s \xrightarrow{\sigma_1} s'' \xrightarrow{\sigma_2} \mathcal{N}' s'$ where s'' contains a token on the edge between x and y , then $s \xrightarrow{\sigma_1 a b \sigma_2} \mathcal{N}' s'$ also holds, where a corresponds to the transition for v and b to the transition for w .

2. By Cor. A.7, if $\llbracket \mathcal{N} \rrbracket(s) = s'$ then the sequence of relations can be written as $s \xrightarrow{\sigma u \sigma'} \mathcal{N}' s'$ where u is the transition by the U-Apply rule for the unitary gate node on the tokens t_1, \dots, t_m . Moreover, considering the fact that u can be reduced, for a token t_j the sequence of relations is $s \xrightarrow{\sigma_{j1} a_j \sigma_{j2} b_j \sigma_{j3} c_j \sigma_{j4} d_j \sigma_{j5} u \sigma_{j6}} \mathcal{N}' s'$ where a_j, b_j, c_j, d_j are transitions shown in the figure in Lem. A.9. By Lem. A.5, $s \xrightarrow{\sigma_{j1} \sigma_{j2} \sigma_{j3} \sigma_{j4} \sigma_{j5} a_j b_j c_j d_j u \sigma_{j6}} \mathcal{N}' s'$ also holds since $\sigma_{j2}, \sigma_{j3}, \sigma_{j4}, \sigma_{j5}$ cannot contain any transition on t_j . Repeating this argument, it can be checked that $s \xrightarrow{\sigma_1 a_1 b_1 c_1 d_1 a_2 b_2 c_2 d_2 \cdots a_m b_m c_m d_m u \sigma_2} \mathcal{N}' s'$ holds. By Lem. A.9 $\bar{U}(s) \xrightarrow{\sigma_1 x_1 x_2 \cdots x_m \sigma_2} \mathcal{N}' s'$ holds, hence $(\llbracket \mathcal{N}' \rrbracket \circ \bar{U})(s) = s'$.

Conversely, if $(\llbracket \mathcal{N}' \rrbracket \circ \bar{U})(s) = s'$, Then by definition there is a relation $s \xrightarrow{\sigma} \mathcal{N}' s'$. By Lem. A.5, $\bar{U}(s) \xrightarrow{a_{11} a_{12} \cdots a_{1m_1} a_{21} \cdots a_{nm_n} x_1 x_2 \cdots x_n \sigma} \mathcal{N}' s'$ holds

where x_j is by new rule and a_{jk} is on the same token as y_j is. By Lem. A.9, $\llbracket \mathcal{N} \rrbracket (s) = s'$.

3. Let v and w be the if node and the new node under reduction respectively. Assume $\llbracket \mathcal{N} \rrbracket (s) = s'$ for $s \in I_{\mathcal{N}}$. By Lem. A.6, a token necessarily visits the edge between v and w in the transitions reaching s' . Hence any sequence of transition relations $s \xrightarrow{\sigma}_{\mathcal{N}} s'$ contains a transition by the if-Meas rule for v . By Lem. A.5, there is a relation $s \xrightarrow{x_1 x_2 x_3 m}_{\mathcal{N}} s'' \xrightarrow{\sigma}_{\mathcal{N}} s'$ where x_1, x_2, x_3 are transitions depicted in Fig. A.1 and m is a transition by the if-Meas rule for v . By definition of if-Meas rule and $\overline{\text{meas}}^v$, it can easily be checked

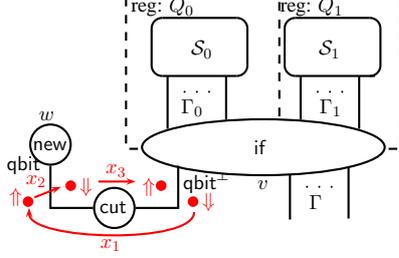


Figure A.1: transitions x_1, x_2, x_3

that $s'' \sim \overline{\text{meas}}^v(s)$, i.e. s'' and $\overline{\text{meas}}^v(s)$ are the same except b . Thus $\overline{\text{meas}}^v(s) \xrightarrow{\sigma}_{\mathcal{N}'} s'_0$ where $s'_0 \sim s'$.

Conversely, if $(\llbracket \mathcal{N}' \rrbracket \circ \overline{\text{meas}}^v)(s) \xrightarrow{\sigma}_{\mathcal{N}'_j} s'_j$, then $s \xrightarrow{x_1 x_2 x_3 m \sigma} s'$ where s'_j is the same as s' except b .

□