A Non-linear Quantum Lambda Calculus

RIKIYA KASHIWAGI and ATSUSHI IGARASHI, Kyoto University, Japan

This paper presents a quantum extension of lambda calculus without linearity, reinterpreting contraction and weakening as logical operations on references to quantum states, rather than on the states themselves. The proposed calculus is a conservative extension of the normal lambda calculus, preserving its operational equivalence. This provides a familiar yet powerful framework for hybrid quantum-classical programming.

1 Introduction

The theoretical foundations of quantum programming languages have largely been built upon linear logic and its variants[1, 6, 7, 9, 10], reflecting the *No-Cloning* Theorem[12] and the *No-Deleting* Theorem[4] of quantum mechanics. Within this paradigm, variables representing quantum states are treated as linear resources that can be neither duplicated (contracted) nor discarded (weakened). While this linearity ensures the physical correctness, it creates a significant gap with the intutions of classical, non-linear programming.

This paper proposes a quantum language designed as a direct quantum extension of the normal (non-linear) lambda calculus. Our central design principle is to ensure this extension is conservative; specifically, β-equivalence of the original non-linear lambda calculus is preserved by the quantum extension. This approach ensures any embedded classical program executes with its standard semantics, making the language's classical fragment indistinguishable from the original lambda calculus. A key consequence of enforcing this conservativity property is that the quantum features of the language are cleanly isolated from the classical host. This quantum-native fragment, consisting of encapsulated unitary operations, forms a distinct module for expressing quantum control. This separation provides a natural framework that enables coexistence of the historically separate paradigms of '*Quantum Data, Classical Control*'[8] and '*Quantum Data, Quantum Control*'[3]. That means a programmer can compose the main logic in a higher-order classical style, while delegating complex quantum subroutines to specialized, domain-specific modules. The resulting language provides a familiar framework for classical programmers and, at the same time, introduces a structured and powerful abstraction for high-level quantum-classical programming.

2 Main idea

To build a quantum language on a non-linear foundation, a reinterpretation of the structural rules—contraction and weakening—is essential. In conventional linear type systems, contraction is viewed as data duplication and weakening as data deletion. This interpretation conflicts with the No-Cloning and No-Deleting theorems when the data is a quantum state.

Our approach reinterprets these rules using *reference-based semantics*. Contraction is not the physical copying of a quantum state, but merely the duplication of a reference to it. Similarly, weakening is simply the act of dropping a reference. As these are purely logical operations on references, they do not violate any physical laws. The physical instantiation of a quantum state occurs only when a unitary operator is applied to one of its duplicated references. Specifically, if the state is expressed as $\sum_i \alpha_i |i\rangle$, where $\{|i\rangle\}$ is a basis set, and a unitary U is applied, the operation transforms the state into the entangled state $\sum_i \alpha_i |i\rangle (U |i\rangle)$. This is similar to the '*Contraction as sharing*' concept like in [1, 2], but it occurs not at the contraction step, but right before the application of a unitary.

Authors' Contact Information: Rikiya Kashiwagi; Atsushi Igarashi, Kyoto University, Graduate School of Informatics, Kyoto, Japan, {kashiwagi,igarashi}@fos.kuis.kyoto-u.ac.jp.

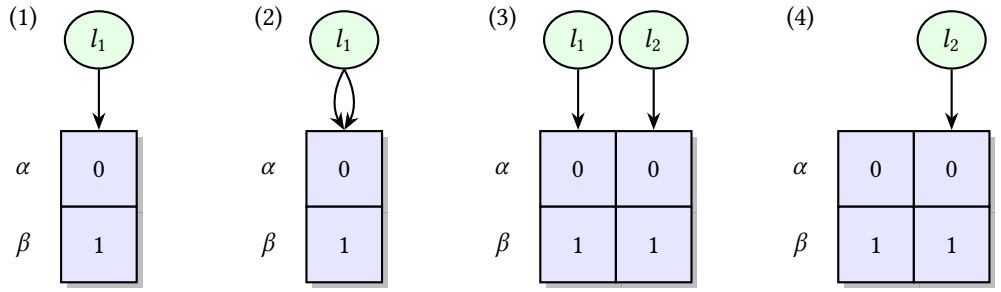


Fig. 1. Illustration of the internal states when $\pi_1 \circ \delta_I$ is applied to a qubit. The grids correspond to quantum states. The rows represent composite systems, and the columns represent superposition with the scalers on the left. From left to right: (1) l_1 refers to some qubit state. (2) Contraction duplicates a reference, not a state. (3) Applying an identity unitary to one of the duplicated references creates a distinct physical instance . (4) Weakening drops a reference.

This design has profound consequences for the language's equational theory. Consider a copy function $\delta \equiv \lambda x.\langle x, x\rangle$ and a projection $\pi_1 \equiv \lambda\langle x, y\rangle.x$. In our system, the equality $\pi_1 \circ \delta = \lambda x.x$ holds. This is because weakening (via π_1) is a logical dereference, not a physical act. In contrast, this identity fails in languages like QML[1] or Qunity[11], where weakening is interpreted as a partial trace. In those systems, the composition $\pi_1 \circ \delta$ induces decoherence, mapping a pure state to a mixed one.

However, this equational property is carefully constrained. If we define a copy function with inserting an identity unitary, $\delta_I \equiv \lambda x.\langle I\, x, x\rangle$, then $\pi_1 \circ \delta_I \neq \lambda x.x$. The application of the unitary I makes $I\, x$ a distinct physical instance from x; for example, the input state $\alpha\,|0\rangle + \beta\,|1\rangle$ becomes the entangled state $\alpha\,|00\rangle + \beta\,|11\rangle$. The evaluation procedure for this case is illustrated in Fig. 1. This behavior does not violate the conservative extension property of our language. Unitary operators like I are separated from the classical fragment. The conservativity applies only to the common language, and equational laws are not expected to hold for terms containing unitaries.

3 The $\Lambda(\mathcal{U})$ calculus

3.1 Syntax

As outlined in the introduction, our language is designed as a conservative extension of a non-linear lambda calculus denoted by Λ. Λ includes lambda terms, the unit and introductions and eliminations of products and sums. The key to our design is the isolation of quantum features from Λ, which is provided by a set $\mathcal{U}$ of external unitary operations.

Definition 3.1 (The surface language $\Lambda(\mathcal{U})$)**.** Given a set of terms $\mathcal{U}$, the language $\Lambda(\mathcal{U})$ is the set of terms defined by the following syntax:

$$\begin{aligned} M, N, P ::= \quad & * \mid x \mid \lambda x.M \mid M\ N \mid \langle M, N\rangle \mid \mathtt{inj}_0\ M \mid \mathtt{inj}_1\ M \mid \\ & \mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N \mid \mathtt{match}\ M\ [x \Rightarrow N \mid y \Rightarrow P] \mid \hat{U} \end{aligned}$$

where $\hat{U} \in \mathcal{U}$.

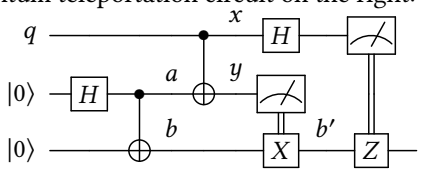
From the perspective of the host language $\Lambda(\cdot)$, $\hat{U}$ is treated as an opaque, first-class value. The language $\mathcal{U}$ can represent any unitary transformation, ranging from a simple unitary gate to a complex subroutine that could be described by a domain-specific language, such as one based on the 'Quantum Data, Quantum Control' paradigm.

Example 3.2. The classical bit can be represented as $i \equiv \mathtt{inj}_i * $ where $i = 0, 1$. A standard conditional expression $\mathtt{if}\ M\ \mathtt{then}\ N\ \mathtt{else}\ P$ can be written as a syntactic sugar for $\mathtt{match}\ M\ [x \Rightarrow N \mid y \Rightarrow P]$, where x and y are fresh variables. Using these notations, the logical NOT is $not \equiv \lambda x.\mathtt{if}\ x\ \mathtt{then}\ 1\ \mathtt{else}\ 0$, and the logical AND is $and \equiv \lambda x.\lambda y.\mathtt{if}\ x\ \mathtt{then}\ y\ \mathtt{else}\ 0$.

The quantum aspect of the language is captured by the $\hat{U}$ term. Match expressions need special attention in this context. It behaves as usual only when the outermost of the scrutinee is evaluated to inl or inr. When it is evaluated to superposition of inl and inr, it performs a measurement to project one of them before that.

Example 3.3 (Quantum teleportation)**.** Let $\mathcal{U}$ be $\{H, CNOT, Z, X\}$ as the standard quantum gate set. The following program implements the quantum teleportation circuit on the right:

$$\begin{array}{l}\mathtt{let}\ \langle a, b\rangle = CNOT\ \langle H\ 0, 0\rangle\ \mathtt{in}\\ \mathtt{let}\ \langle x, y\rangle = CNOT\ \langle q, a\rangle\ \mathtt{in}\\ \mathtt{let}\ b' = \mathtt{if}\ y\ \mathtt{then}\ X\ b\ \mathtt{else}\ b\ \mathtt{in}\\ \mathtt{if}\ H\ x\ \mathtt{then}\ Z\ b'\ \mathtt{else}\ b'\end{array}$$



where $\mathtt{let}\ x = M\ \mathtt{in}\ N$ is a syntactic sugar for $(\lambda x.N)\ M$.

It should be noted that the *not* function in Example 3.2 does not behave as the X gate (or the quantum NOT gate) when applied to a qubit, because the input is measured in the condition of the if expression, which collapses any superposition.

As a powerful example of this encapsulation of $\mathcal{U}$, a syntax inspired by pattern isomorphism[7] allows for the direct declaration of unitary operations as follows. Let $\mathcal{U}$ be defined using the pattern isomorphism syntax. For instance, an X gate can be defined as $X \equiv \{0 \leftrightarrow 1 \mid 1 \leftrightarrow 0\}$.

3.2 Mathematical preliminaries for operational semantics

Our operational semantics relies on two key mathematical structures: ℓ^2 *space* to represent quantum states, and *finite distributions* to model probabilistic reduction.

Definition 3.4 (ℓ^2 space)**.** Given a countable set X, define ℓ^2 space as

$$\ell^2(X) := \left\{\psi : X \to \mathbb{C} \;\middle|\; \|\psi\| := \sum\nolimits_{x\in X} |\psi(x)|^2 < \infty\right\},$$

and inner product $\langle\psi, \phi\rangle = \sum_{x\in X} \overline{\psi(x)}\phi(x)$. It is known that $(\ell^2(X), \langle\cdot,\cdot\rangle)$ forms a Hilbert space.

Definition 3.5 (Finite distributions)**.** Given a countable set X, define a set

$$\mathcal{D}(X) := \left\{\mu : X \to\ [0,1] \;\middle|\; |\mu| := \sum\nolimits_{x\in X} \mu(x) = 1,\ |\mathrm{supp}(\mu)| < \infty\right\}.$$

Any element of $\ell^2(X)$ and $\mathcal{D}(X)$ can be written as linear combinations: $\psi = \sum_{x\in X} \psi(x)\delta_x$ and $\mu = \sum_{x\in X} \mu(x)\delta_x$ respectively, where δ_x is defined by $\delta_x(y) = 1$ if $y = x$ and 0 otherwise.

Next, we utilize two linear map constructions for ℓ^2. The first is the functorial action of ℓ^2, which lifts any function $f : X \to Y$ to a linear map $\ell^2(f) : \ell^2(X) \to \ell^2(Y)$ defined by $\ell^2(f)(\delta_x) := \delta_{f(x)}$. The second one extends a function $k : X \to \ell^2(Y)$ to a linear map $k^\sharp : \ell^2(X) \to \ell^2(Y)$ via linear extension: $k^\sharp(\sum_{x\in X} \alpha_x\delta_x) := \sum_{x\in X} \alpha_x k(x)$ only if the resulting sum converges for any input.

3.3 Operational semantics

This section defines the operational semantics, describing how the reference-based semantics from Section 2 is formalized. The state of a computation is captured by a *configuration*: a pair of a global store and the term being evaluated. Crucially, the store holds the physical quantum state, while the

term operates only on locations ($\mathcal{L}$)—references to that store. This separation allows contraction and weakening to be treated as logical operations on references, without directly affecting the physical quantum state. Because a term is rewritten to contain these locations during execution, we must first define a distinct runtime language that extends the surface syntax with them.

Definition 3.6 (The runtime language $\Lambda_{\mathcal{L}}(\mathcal{U})$)**.** Given a set of store locations $\mathcal{L}$, the runtime language is defined as $\Lambda_{\mathcal{L}}(\mathcal{U}) := \Lambda(\mathcal{U} \cup \mathcal{L})$. To simplify notation, M, N, P will also range over runtime terms. We will explicitly refer to surface terms when the distinction is significant.

Only first-order, or *basis values*, can be placed in the store and exist in a superposition. Values and basis values are given by the following syntax.

$$\begin{array}{lll} \text{Values} & V, W ::= & l \mid * \mid \lambda x.M \mid \langle V, W\rangle \mid \mathtt{inj}_0\, V \mid \mathtt{inj}_1\, V \mid \hat{U} \\ \text{Basis values} & \hat{V}, \hat{W} ::= & * \mid \langle \hat{V}, \hat{W}\rangle \mid \mathtt{inj}_0\, \hat{V} \mid \mathtt{inj}_1\, \hat{V} \end{array}$$

The set of all basis values is denoted by $\mathcal{V}_0$.

A configuration $[\sigma, M]$ represents the state of a computation. The quantum store σ is a vector in a free Hilbert space over value stores (finite maps from locations $\mathcal{L}$ to basis values $\mathcal{V}_0$), while M is the runtime term containing references to σ. Each value store in the support of σ must have the same domain; we can define $\mathrm{dom}(\sigma)$ without ambiguity. Write C for the set of all configurations.

The operational semantics are defined by a call-by-value, small-step reduction relation ($\longrightarrow$). The key rules in Fig. 2 directly correspond to the four postulates of quantum mechanics[5].

State preparation (Q-Prep) It prepares a quantum state by taking a classical basis value $\hat{V}$ and placing it into the store at a new location l.

Unitary Evolution (Q-Evolve) Applying a unitary $\hat{U}$ to a reference l_1 transforms the entire quantum store σ according to the operator's interpretation. The location l_1 stays in the store, which reflects the 'Contraction as sharing' mechanism as illustrated in Fig. 1.

Measurement (Q-Meas) The match expression performs a measurement on a referenced state, causing the computation to branch probabilistically and collapsing the quantum store.

Composite system (Q-Destr) It destructs a pair stored at l_1 and creates new, distinct references l_2, l_3 to its components. It copies values at each value store, not meaning that the superpositions themselves are cloned.

This semantics imposes a meta-level constraint on $\hat{U}$. Specifically, the *interpretation* $[\![\hat{U}]\!]_{\mathcal{U}} : \mathcal{V}_0 \rightharpoonup \ell^2(\mathcal{V}_0)$ must satisfy the following condition: if $U = [\![\hat{U}]\!]_{\mathcal{U}}$, then $(U|_{\mathrm{dom}(U)})^\sharp$ is unitary.

Definition 3.7 (Small-step reduction)**.** Given a set of partial functions $[\![\hat{U}]\!]_{\mathcal{U}}$ for $\hat{U} \in \mathcal{U}$, the reduction relation $\longrightarrow \subseteq C \times \mathcal{D}(C)$ is the smallest relation closed under the classical rules (Appendix A.1) and the quantum rules (Fig. 2). The relation is then extended to all evaluation contexts E via the standard congruence rule: if $[\sigma, M] \longrightarrow \mu$, then $[\sigma, E[M]] \longrightarrow \sum_{[\sigma', M'] \in \mathrm{supp}(\mu)} \mu([\sigma', M'])\delta_{[\sigma', E[M']]}$. The definition of evaluation contexts is given in Appendix A.1.

A key property to the physical realizability of the semantics is norm preservation, which guarantees that the norm of the quantum state does not change during reduction.

Theorem 3.8 (Norm preservation)**.** *If* $[\sigma, M] \longrightarrow \mu$*, then for all* $[\sigma', M'] \in \mathrm{supp}(\mu)$, $||\sigma'|| = ||\sigma||$.

Since a single reduction step can yield a distribution of configurations, the *multi-step reduction* ($\longrightarrow^*$) is not a simple reflexive and transitive closure. Instead, it is defined inductively to compose these probabilistic outcomes, akin to a monadic bind.

Definition 3.9 (Multi-step reduction)**.** The multi-step reduction relation $\longrightarrow^* \subseteq C \times \mathcal{D}(C)$ is the smallest relation satisfying:

$$\frac{f(\rho) = \rho \cup \{(l_2, \hat{V}), (l_3, \hat{W})\} \text{ where } \rho(l_1) = \langle \hat{V}, \hat{W} \rangle \qquad l_2, l_3 \notin \mathrm{dom}(\sigma)}{[\sigma, \mathtt{let}\ \langle x, y \rangle = l_1\ \mathtt{in}\ M] \longrightarrow \delta_{[\ell^2(f)(\sigma), M[l_2/x, l_3/y]]}} \text{Q-Destr}$$

$$\frac{\begin{array}{c}|\alpha_0|^2 + |\alpha_1|^2 = 1 \qquad ||\sigma_0|| = ||\sigma_1|| \qquad f_0(\rho) = \rho \cup \{(l_2, \hat{V})\} \text{ where } \rho(l_1) = \mathtt{inj}_0\ \hat{V} \\ f_1(\rho) = \rho \cup \{(l_3, \hat{W})\} \text{ where } \rho(l_1) = \mathtt{inj}_1\ \hat{W} \qquad l_2 \notin \mathrm{dom}(\sigma_0) \qquad l_3 \notin \mathrm{dom}(\sigma_1)\end{array}}{\begin{array}{c}[\alpha_0\sigma_0 + \alpha_1\sigma_1, \mathtt{match}\ l_1\ [y \Rightarrow M \mid z \Rightarrow N]] \\ \longrightarrow |\alpha_0|^2 \delta_{[\ell^2(f_0)(\sigma_0), M[l_2/y]]} + |\alpha_1|^2 \delta_{[\ell^2(f_1)(\sigma_1), N[l_3/z]]}\end{array}} \text{Q-Meas}$$

$$\frac{f(\rho) = \rho \cup \{(l, \hat{V})\} \qquad l \notin \mathrm{dom}(\sigma)}{[\sigma, \hat{U}\ \hat{V}] \longrightarrow \delta_{[\ell^2(f)(\sigma), \hat{U}\ l]}} \text{Q-Prep} \qquad \frac{f(\rho) = \sum_{\hat{V}} [\![\hat{U}]\!] u(\rho(l_1))(\hat{V}) \delta_{\rho \cup \{(l_2, \hat{V})\}} \qquad l_2 \notin \mathrm{dom}(\sigma)}{[\sigma, \hat{U}\ l_1] \longrightarrow \delta_{[f^\sharp(\sigma), l_2]}} \text{Q-Evolve}$$

Fig. 2. Quantum reduction rules

- $[\sigma, M] \longrightarrow^* \delta_{[\sigma, M]}$ for all $[\sigma, M] \in C$.
- If $[\sigma, M] \longrightarrow \mu$ and for all $[\sigma', M'] \in \mathrm{supp}(\mu)$, $[\sigma', M'] \longrightarrow^* \nu_{\sigma', M'}$, then $[\sigma, M] \longrightarrow \sum_{[\sigma', M'] \in \mathrm{supp}(\mu)} \mu([\sigma', M'])\nu_{\sigma', M'}$.

Then, we define an equivalence-like relation based on multi-step reduction.

Definition 3.10. For all $M, N \in \Lambda$, define $M \longleftrightarrow^* N$ iff for any σ, there exist μ such that $[\sigma, M] \longrightarrow^* \mu$ and $[\sigma, N] \longrightarrow^* \mu$.

3.4 Conservativity

As mentioned in Section 1, the central design principle of our language is *conservativity*; a formal guarantee that the quantum extension does not alter the original behavior of the classical fragment. Informally, this means that for any term, one may apply the classical reduction rules to the subterms that do not contain $\hat{U}$ without changing the overall behavior of the term.

Conservativity is then stated as the following theorem. The proof is given in Appendix B.

Theorem 3.11 (Conservativity)**.** *For all $M, N \in \Lambda$, $M \longleftrightarrow^*_\Lambda N$ iff $M \longleftrightarrow^* N$.*

This property cannot be proved for general operational equivalence other than β-equivalence since our language yields side-effects through measurement. Actually, η-equivalence of `match` does not hold in general. Consider the term $\mathtt{match}\ x\ [y \Rightarrow \mathtt{inj}_0\ y \mid z \Rightarrow \mathtt{inj}_1\ z]$. In classical reduction, it is η-equivalent to x. However, x can store a quantum state, and thus measuring it results in a probabilistic outcome.

4 Future work

We aim to develop both practical compilation techniques and theoretical extensions. An intermediate language will be designed to explicitly describe interactions between classical and quantum systems through lower-level quantum operations. Although the simply typed system is presented in Appendix C, it can be extended with polymorphism and recursive types. We are also working on a denotational semantics to formalize the language's mathematical interpretations.

References

[1] T. Altenkirch and J. Grattage. 2005. A Functional Quantum Programming Language. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*. IEEE, Chicago, IL, USA, 249–258. doi:10.1109/LICS.2005.1

[2] Pablo Arrighi and Gilles Dowek. 2004. Operational Semantics for Formal Tensorial Calculus. In *Proceedings of the International Workshop on Quantum Programming Languages*. 21–38.
[3] Alejandro Díaz-Caro. 2022. A Quick Overview on the Quantum Control Approach to the Lambda Calculus. *Electronic Proceedings in Theoretical Computer Science* 357 (April 2022), 1–17. doi:10.4204/EPTCS.357.1 arXiv:2204.03885 [cs]
[4] Arun Kumar Pati and Samuel L. Braunstein. 2000. Impossibility of Deleting an Unknown Quantum State. *Nature* 404, 6774 (March 2000), 164–165. doi:10.1038/404130b0
[5] Michael A. Nielsen and Isaac L. Chuang. 2010. Quantum Computation and Quantum Information: 10th Anniversary Edition. https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE. doi:10.1017/CBO9780511976667
[6] Neil J. Ross. 2017. Algebraic and Logical Methods in Quantum Computation. doi:10.48550/arXiv.1510.02198 arXiv:1510.02198 [quant-ph]
[7] Amr Sabry, Benoît Valiron, and Juliana Kaizer Vizzotto. 2018. From Symmetric Pattern-Matching to Quantum Control. In *Foundations of Software Science and Computation Structures*, Christel Baier and Ugo Dal Lago (Eds.). Springer International Publishing, Cham, 348–364. doi:10.1007/978-3-319-89366-2_19
[8] Peter Selinger. 2004. Towards a Quantum Programming Language. *Mathematical Structures in Computer Science* 14, 4 (Aug. 2004), 527–586. doi:10.1017/S0960129504004256
[9] Peter Selinger and Benoît Valiron. 2009. Quantum Lambda Calculus. In *Semantic Techniques in Quantum Computation*. Cambridge University Press, 135–172. doi:10.1017/CBO9781139193313.005
[10] André van Tonder. 2004. A Lambda Calculus for Quantum Computation. *SIAM J. Comput.* 33, 5 (Jan. 2004), 1109–1135. doi:10.1137/S0097539703432165
[11] Finn Voichick, Liyi Li, Robert Rand, and Michael Hicks. 2023. Qunity: A Unified Language for Quantum and Classical Computing. *Proceedings of the ACM on Programming Languages* 7, POPL (Jan. 2023), 921–951. doi:10.1145/3571225
[12] W. K. Wootters and W. H. Zurek. 1982. A Single Quantum Cannot Be Cloned. *Nature* 299, 5886 (Oct. 1982), 802–803. doi:10.1038/299802a0

A Operational semantics

A.1 Full definitions of operational semantics

The remaining part of the single-step reduction relation (Definition 3.7) is the classical β-reduction rules, given in Fig. 3. These rules are standard for call-by-value lambda calculus, and do not interact with the quantum store.

$$(\lambda x.M)\ V \longrightarrow_\Lambda M[V/x]\ \beta\text{-}\textsc{Lam} \qquad \texttt{let}\ \langle x, y\rangle = \langle V, W\rangle\ \texttt{in}\ M \longrightarrow_\Lambda, M[V/x, W/y]\ \beta\text{-}\textsc{Pair}$$

$$\texttt{match}\ \texttt{inj}_0\ V\ [x \Rightarrow M \mid y \Rightarrow N] \longrightarrow_\Lambda M[V/x]\ \beta\text{-}\textsc{Inj}_0$$

$$\texttt{match}\ \texttt{inj}_1\ V\ [x \Rightarrow M \mid y \Rightarrow N] \longrightarrow_\Lambda N[V/y]\ \beta\text{-}\textsc{Inj}_1$$

Fig. 3. Classical reduction rules

The relation $\longrightarrow$ is defined by the rules in Fig. 2 and Fig. 3, together with the congruence rule induced by the evaluation contexts. They are defined as:

$$\begin{aligned} E ::= \quad & [\cdot] \mid E\ M \mid V\ E \mid \langle E, M\rangle \mid \langle V, E\rangle \mid \texttt{inj}_0\ E \mid \texttt{inj}_1\ E \mid \\ & \texttt{let}\ \langle x, y\rangle = E\ \texttt{in}\ M \mid \texttt{match}\ E\ [x \Rightarrow M \mid y \Rightarrow N] \end{aligned}$$

A.2 Proof of norm preservation

Lemma A.1 (Linearity of norm). *$\| \sum_i \alpha_i \psi_i \| = \sqrt{\sum_i |\alpha_i|^2 \|\psi_i\|^2}$ if $\langle \psi_i, \psi_j \rangle = 0$ for all $i \neq j$.*

Lemma A.2. *For all injection $f : X \to Y$ and $\psi \in \ell^2(X)$, $\|\ell^2(f)(\psi)\| = \|\psi\|$.*

Proof. Let g be a left inverse of f, then

$$\|\ell^2(f)(\psi)\| = \left\|\sum_{x\in X}\psi(x)\delta_{f(x)}\right\| = \sum_{y\in f(X)}|\psi(g(y))|^2 = \sum_{x\in X}|\psi(x)|^2 = \|\psi\|. \tag{1}$$

□

Lemma A.3. *For all isometry $f : \ell^2(X) \to \ell^2(Y)$ and $\psi \in \ell^2(X)$, $\|f(\psi)\| = \|\psi\|$.*

Proof. For all $x, x' \in X$, $\langle f(\delta_x), f(\delta_{x'})\rangle = \langle \delta_x, \delta_{x'}\rangle = 0$ holds because f is an isometry. Then, by Lemma A.1,

$$\|f(\psi)\| = \left\|\sum_{x\in X}\psi(x)f(\psi)\right\| = \sqrt{\sum_{x\in X}|\psi(x)|^2\|f(\delta_x)\|^2}.$$

As $\|f(\delta_x)\| = \sqrt{\langle f(\delta_x), f(\delta_x)\rangle} = \|\delta_x\| = 1$, we have $\|f(\psi)\| = \sqrt{\sum_{x\in X}|\psi(x)|^2} = \|\psi\|$. □

Theorem 3.8 (Norm preservation). *If $[\sigma, M] \longrightarrow \mu$, then for all $[\sigma', M'] \in \mathrm{supp}(\mu)$, $||\sigma'|| = ||\sigma||$.*

Proof. By induction on the derivation of $[\sigma, M] \longrightarrow \mu$. We only show the cases for the quantum rules; the other cases are immediate.

- Case Q-Destr: Since the function f adds new entries l_2 and l_3 to ρ, it is injective; hence by Lemma A.2, $\|\ell^2(f)(\sigma)\| = \|\sigma\|$.
- Case Q-Prep: The same reasoning as in Case Q-Destr.
- Case Q-Meas: Similar argument to the Case Q-Destr, f_0 and f_1 are injective. For each basis ρ_i in σ_i $(i = 0, 1)$, $\rho_0 \neq \rho_1$ holds because they differ at least on the values stored in l_1; thus, $\langle\sigma_0, \sigma_1\rangle = 0$.

$$\begin{aligned}
\|\alpha_0\sigma_0 + \alpha_1\sigma_1\| &= \sqrt{|\alpha_0|^2\|\sigma_0\|^2 + |\alpha_1|^2\|\sigma_1\|^2} && \text{(by Lemma A.1)}\\
&= \sqrt{(|\alpha_0|^2 + |\alpha_1|^2)\|\sigma_0\|^2} && (\|\sigma_0\| = \|\sigma_1\|)\\
&= \|\sigma_0\| && (|\alpha_0|^2 + |\alpha_1|^2 = 1)\\
&= \|\ell^2(f_0)(\sigma_0)\| && \text{(by Lemma A.2)}
\end{aligned}$$

  Similarly, $\|\alpha_0\sigma_0 + \alpha_1\sigma_1\| = \|\ell^2(f_1)(\sigma_1)\|$.
- Case Q-Evolve: As $(\llbracket\hat{U}\rrbracket_{\mathcal{U}})^\sharp$ is a unitary, $\|\llbracket\hat{U}\rrbracket_{\mathcal{U}}(\rho(l_1))\|^2 = 1$; hence, $\|f(\rho)\| = 1$ for any basis ρ in σ. For all basis ρ, ρ' in σ,

$$\langle f(\rho), f(\rho')\rangle = \sum_{\hat{V}}\overline{\llbracket\hat{U}\rrbracket_{\mathcal{U}}(\rho(l_1))(\hat{V})}\llbracket\hat{U}\rrbracket_{\mathcal{U}}(\rho'(l_1))(\hat{V}) = \langle\delta_\rho, \delta_{\rho'}\rangle.$$

  As $f^\sharp(\delta_\rho) = f(\rho)$ by definition, $\langle f^\sharp(\delta_\rho), f^\sharp(\delta_{\rho'})\rangle = \langle\delta_\rho, \delta_{\rho'}\rangle$ holds, which means $f^\sharp$ is an isometry. Thus, by Lemma A.3, $\|f(\sigma)\| = \|\sigma\|$.

□

B Proof of conservativity

Lemma B.1. *For any $M \in \Lambda$, σ and μ, if $[\sigma, M] \longrightarrow^* \mu$ holds, then there exists $M' \in \Lambda$ such that $M \longrightarrow_\Lambda^* M'$.*

Proof. First, we prove that if $[\sigma, M] \longrightarrow \mu$ then $\exists M'.M \longrightarrow_\Lambda M'$ by induction on the derivation of $M \longrightarrow \mu$.

- Case β rules: The same rule applies to $\longrightarrow_\Lambda$ and $\mu = \delta_{[\delta_\emptyset, M']}$.
- Case Q rules: Impossible since each M includes a unitary or locations, meaning $M \notin \Lambda$.

Then, the original statement is proved by simple induction on the derivation of $M \longrightarrow^* \mu$. □

Church-Rosser and confluence properties for the classical reduction directly follows from the determinism of $\longrightarrow_\Lambda$.

Lemma B.2 (Church-Rosser for classical reduction). *For any $M, N \in \Lambda$, if $M \longleftrightarrow^*_\Lambda N$ holds, then there exists $P \in \Lambda$ such that $M \longrightarrow^*_\Lambda P$ and $N \longrightarrow^*_\Lambda P$.*

Lemma B.3 (Confluence for classical reduction). *For any $M, N_1, N_2 \in \Lambda$, if $M \longrightarrow_\Lambda N_1$ and $M \longrightarrow_\Lambda N_2$ hold, then there exists $P \in \Lambda$ such that $N_1 \longrightarrow^*_\Lambda P$ and $N_2 \longrightarrow^*_\Lambda P$.*

Theorem 3.11 (Conservativity). *For all $M, N \in \Lambda$, $M \longleftrightarrow^*_\Lambda N$ iff $M \longleftrightarrow^* N$.*

Proof. ($\Rightarrow$) By Lemma B.2, there exists $P \in \Lambda$ such that $M \longrightarrow^*_\Lambda P$ and $N \longrightarrow^*_\Lambda P$. Then, by defition of $\longrightarrow$, we have $[\delta_\emptyset, M] \longrightarrow^* \delta_{[\delta_\emptyset, P]}$ and $[\delta_\emptyset, N] \longrightarrow^* \delta_{[\delta_\emptyset, P]}$, which implies $M \longleftrightarrow^* N$. ($\Leftarrow$) By the definition of $\longleftrightarrow^*$, there exists μ such that $[\delta_\emptyset, M] \longrightarrow^* \mu$ and $[\delta_\emptyset, N] \longrightarrow^* \mu$. By Lemma B.1, there exists $M', N' \in \Lambda$ such that $M \longrightarrow^*_\Lambda M'$ and $N \longrightarrow^*_\Lambda N'$. By Lemma B.2, there exists $P \in \Lambda$ such that $M' \longrightarrow^*_\Lambda P$ and $N' \longrightarrow^*_\Lambda P$; hence, we have $M \longleftrightarrow^*_\Lambda N$. □

C Type system

We present the simple type system of $\Lambda(\mathcal{U})$ calculus. Syntax of types and typing context is given as follows.

Types	$A, B, C ::=$	$\top \mid A \otimes B \mid A \oplus B \mid A \to B$
Basis types	$\hat{A}, \hat{B} ::=$	$\top \mid \hat{A} \otimes \hat{B} \mid \hat{A} \oplus \hat{B}$
Typing context	$\Gamma ::=$	$\epsilon \mid \Gamma, x : A$

The types include the unit type $\top$, product types $A \otimes B$, sum types $A \oplus B$, and function types $A \to B$. $\otimes$ and $\oplus$ indicate that they are tensor product and direct sum of Hilbert spaces when interpreted as superpositions. Crucially, there is no distinction between classical and quantum types in this system. For instance, the type of both classical bit and qubit are represented as the same $\top \oplus \top$.

Typing judgements are defined on the runtime terms, which means they include typing rules for store locations. In order to map locations to their types, we introduce the notion of store typing.

Definition C.1 (Store typing). A store typing Σ is a finite mapping from locations to basis types. A store σ is well-typed under Σ, denoted $\vdash \sigma : \Sigma$, if for every $l \in \mathrm{dom}(\Sigma)$, $\vdash \sigma(l) : \Sigma(l)$ holds.

Similar to the definition of the reduction relation, we assume a typing judgement for $\hat{U} \in \mathcal{U}$ in the form of $\vdash_{\mathcal{U}} \hat{U} : \hat{A} \to \hat{B}$.

Definition C.2 (Typing judgments). Given a set of external typing judgements $\vdash_{\mathcal{U}} \hat{U} : \hat{A} \to \hat{B}$ for $\hat{U} \in \mathcal{U}$, A typing judgment is of the form $\Sigma \mid \Gamma \vdash M : A$. The typing rules are given in Fig. 4.

We can show the standard type safety properties, i.e., subject reduction and progress.

Theorem C.3 (Subject reduction). *If $\Sigma \mid \Gamma \vdash M : A$, $[\sigma, M] \to [\sigma', M']$ and $\vdash \sigma : \Sigma$, then there exists Σ' such that $\Sigma' \mid \Gamma \vdash M^{*'} : A$ and $\vdash \sigma' : \Sigma'$.*

Theorem C.4 (Progress). *If $\Sigma \mid \Gamma \vdash M : A$, then either M is a value or there exists μ such that $[\sigma, M] \to \mu$.*

$$\frac{l \in \mathrm{dom}(\Sigma)}{\Sigma \mid \Gamma \vdash l : \Sigma(l)}\ \textsc{Loc} \qquad \frac{}{\Sigma \mid \Gamma \vdash * : \top}\ \top\text{-I} \qquad \frac{}{\Sigma \mid \Gamma, x : A \vdash x : A}\ \pi_1 \qquad \frac{\Sigma \mid \Gamma \vdash x : A}{\Sigma \mid \Gamma, y : B \vdash x : A}\ \pi_2$$

$$\frac{\Sigma \mid \Gamma, x : A \vdash M : B}{\Sigma \mid \Gamma \vdash \lambda x.M : A \to B}\ \to\text{-I} \qquad \frac{\Sigma \mid \Gamma \vdash M : A \to B \qquad \Sigma \mid \Gamma \vdash N : A}{\Sigma \mid \Gamma \vdash M\ N : B}\ \to\text{-E}$$

$$\frac{\Sigma \mid \Gamma \vdash M : A \qquad \Sigma \mid \Gamma \vdash N : B}{\Sigma \mid \Gamma \vdash \langle M, N \rangle : A \otimes B}\ \otimes\text{-I} \qquad \frac{\Sigma \mid \Gamma \vdash M : A}{\Sigma \mid \Gamma \vdash \mathtt{inj}_0\ M : A \oplus B}\ \oplus_0\text{-I}$$

$$\frac{\Sigma \mid \Gamma \vdash M : B}{\Sigma \mid \Gamma \vdash \mathtt{inj}_1\ M : A \oplus B}\ \oplus_1\text{-I} \qquad \frac{\vdash_{\mathcal{U}} \hat{U} : \hat{A} \to \hat{B}}{\Sigma \mid \Gamma \vdash \hat{U} : \hat{A} \to \hat{B}}\ \textsc{Op}$$

$$\frac{\Sigma \mid \Gamma \vdash M : A \otimes B \qquad \Sigma \mid \Gamma, x : A, y : B \vdash N : C}{\Sigma \mid \Gamma \vdash \mathtt{let}\ \langle x, y \rangle = M\ \mathtt{in}\ N : C}\ \otimes\text{-E}$$

$$\frac{\Sigma \mid \Gamma \vdash M : A \oplus B \qquad \Sigma \mid \Gamma, x : A \vdash N : C \qquad \Sigma \mid \Gamma, y : B \vdash P : C}{\Sigma \mid \Gamma \vdash \mathtt{match}\ M\ [x \Rightarrow N \mid y \Rightarrow P] : C}\ \oplus\text{-E}$$

Fig. 4. Typing rules