Research Article

AND DATA SCIENCES
BULLETIN OF COMPUTER

A Linear Sup-Calculus for Quantum Measurement

Gilles Dowek

École Normale Supérieure Paris-Saclay, France

Abstract

Non-harmonious logical connectives have recently been shown to provide a natural proof-theoretic account of quantum measurement. In particular, the connective sup introduced by Díaz-Caro and Dowek endows intuitionistic natural deduction with both harmonious and excessive rules, and its proof language forms the core of a quantum programming language. However, the resulting calculus is still permissive enough to express non-linear or genuinely non-physical operations: nothing in the type system enforces linearity or prevents the cloning of quantum data. In this paper we introduce a *linear* version of the sup-calculus. We separate classical and quantum data by means of a two-zone context discipline, and we restrict the use of the sup-connective and scalar combinations to ensure that well-typed quantum programs denote linear maps between finite-dimensional Hilbert spaces. We give a natural deduction style type system, an operational semantics with both deterministic and measurement steps, and we prove subject reduction and type soundness. We then show that no term in the purely quantum fragment can implement a duplicator $\mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$, while classical outcomes of measurements can still be duplicated freely. Finally, we sketch a denotational semantics in which closed quantum terms denote completely positive maps, and we relate our system to existing quantum λ-calculi.

Keywords: linear Sup-calculus, quantum lambda calculus, proof theory of quantum measurement, linear type systems, denotational semantics of quantum computation

1. Introduction

The proof-theoretic notion of *harmony* between introduction and elimination rules has played a central role in the analysis of logical connectives, particularly in the tradition stemming from Gentzen and Prawitz [1–3]. Roughly speaking, a harmonious connective is one whose elimination rules act as a faithful inversion of its introduction rules, so that no information is implicitly created or lost in normalisation. In contrast, *non-harmonious* connectives, such as Prior's infamous *tonk*, admit rules that are either insufficient or excessive with respect to this inversion principle [4, 5], and thus break the tight correspondence between proofs and their computational content.

Recent work has argued that excessive connectives provide a natural logical account of phenomena where information is intrinsically lost or irreversibly consumed, most notably quantum measurement [6, 7]. Díaz-Caro and Dowek introduce an intuitionistic propositional logic with a non-harmonious connective sup (read: *sup*, for *superposition*) and a corresponding proof term language, the sup-calculus [8]. In their system, the introduction rules for sup behave like those of conjunction, while its elimination rules partly resemble those of disjunction, capturing the "and vs. or" dual nature of

Corresponding author: Gilles Dowek (gilles.dowek@ens-cachan.fr)

BCDS 2024, Vol. 5(4), 12-32. DOI: 10.71448/bcds2454-2

quantum superposition and measurement. When the proof language is extended with scalars and finite sums, proofs can be interpreted as linear combinations of states and operators [9, 10], yielding a core quantum programming language able to express standard quantum algorithms.

However, as explicitly acknowledged in that work, the resulting calculus remains rather permissive. Although proofs with scalars resemble linear combinations in a Hilbert-space semantics, the type system by itself does not prevent non-linear or non-physical constructions. In particular, it does not exclude inhabitants of types that violate fundamental quantum principles, such as a duplicator $\mathsf{qbit} \Rightarrow \mathsf{qbit} \otimes \mathsf{qbit}$ for genuinely quantum data. The logical insight that non-harmonious rules model measurement is therefore not yet matched by a type-theoretic guarantee that well-typed programs respect the linear structure of quantum mechanics and the constraints expressed by the no-cloning and no-deleting theorems [11, 12].

In this paper we refine the sup-calculus with a *linear type discipline* that enforces the linearity of quantum programs and cleanly separates quantum data from classical information, in the spirit of linear logic and quantum λ-calculi [13–15]. Typing judgements are equipped with a two-zone context: a classical context, where weakening and contraction are permitted and values such as measurement outcomes may be duplicated or discarded freely, and a quantum context, where assumptions must be used exactly once, reflecting the no-cloning and no-deleting principles. Within this setting we extend the original proof language with typing rules for the standard linear connectives and for the sup connective, together with algebraic constructs such as scalars and finite sums, and we adapt the excessive elimination rules of sup to the linear regime so that they continue to model quantum measurement. We define an operational semantics that combines deterministic β-reduction with measurement steps corresponding to these excessive rules and prove subject reduction, ensuring that reduction preserves types and that well-typed closed terms do not "go wrong". On top of this, we show that in the purely quantum fragment of the language there is no term of type $\mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$, so the calculus embodies a no-cloning theorem at the level of types, while classical outcomes of measurements remain duplicable in the unrestricted part of the context. Finally, we sketch a denotational semantics in which types are interpreted as finite-dimensional Hilbert spaces and well-typed terms as completely positive maps [16, 17], and the sup connective as a direct sum with its excessive elimination rules interpreted as quantum measurements, thereby linking the proof-theoretic behaviour of the calculus to standard semantic models of quantum computation.

The rest of the paper is structured as follows. Section 2 recalls the original sup-logic and basic linear typing ideas. Section 3 introduces the syntax and type system of the linear sup-calculus. Section 4 presents the operational semantics and proves subject reduction. Section 5 establishes linearity properties, in particular a no-cloning theorem. Section 6 sketches a denotational semantics. Section 8 discusses related work, and Section 9 concludes.

2. Background

In this section we briefly recall the proof-theoretic viewpoint on harmony, the role of the non-harmonious connective sup, and the basic ideas behind linear type disciplines for quantum data. The presentation is intentionally informal and serves only to situate our work; full technical details can be found in the original sources [1–3].

2.1. Harmony, insufficiency and excessiveness

In natural deduction, each connective $\triangle$ is specified by a collection of introduction and elimination rules. Informally, the introduction rules explain how to construct proofs of formulae of the form $A \triangle B$, while the elimination rules explain how such proofs may be used. Gentzen's inversion principle together with Prawitz's subformula analysis motivate the idea that, for a well-behaved connective, the elimination rules should be a kind of mirror image of the introduction rules: they should not permit one to derive more information than what was made available by an introduction [1, 2, 5]. When this inversion principle is satisfied, the connective is said to be *harmonious*. Dummett and others have argued that harmony is closely tied to the justification of logical laws in proof-theoretic semantics [3, 18].

Connectives that fail this condition do so in two qualitatively different ways. If the elimination rules are too weak to recover all the information encapsulated by the introduction rules, the connective is called *insufficient*. If, on the contrary, the elimination rules are too strong and allow one to exploit a proof in ways that are not justified by how it was introduced, the connective is called *excessive* [5, 19]. In an excessive connective, an elimination rule may implicitly discard part of the information that was present in the introduction; at the level of proof reduction this corresponds to normalisation steps in which some branch or resource disappears without being fully accounted for. Prior's connective *tonk* is the paradigmatic example of such excess [4], but more controlled forms of excessiveness have been studied and can be given a meaningful interpretation in proof-theoretic semantics [7, 20].

2.2. The sup*-connective and quantum measurement*

Building on this perspective, Díaz-Caro and Dowek introduce an intuitionistic propositional logic with the usual connectives $\top$, $\bot$, $\Rightarrow$, $\wedge$, $\vee$ and an additional connective $A \,\mathsf{sup}\, B$ [8]. The introduction rule for sup is identical to that of conjunction: to prove $A \,\mathsf{sup}\, B$ one simultaneously establishes A and B. Its elimination rules, however, partly follow those of disjunction and include an explicitly excessive rule that embodies information loss. The associated proof term language, the sup-calculus, can be extended with scalars and finite sums so that proofs are interpreted as algebraic linear combinations of terms, in the style of algebraic λ-calculi [9, 10].

The central observation is that the pair *superposition / measurement* in quantum computation mirrors the pair *introduction / elimination* for a non-harmonious connective [6, 21]. Preparing a superposition behaves like a conjunction-style introduction: one has simultaneous access to several components. Measuring that superposition behaves like a disjunction-style elimination that irreversibly selects one outcome and discards information about the others. The sup-calculus formalises this analogy: introduction into $A \,\mathsf{sup}\, B$ corresponds to building a joint quantum state, while excessive elimination rules correspond to measurement operations that collapse the state and throw away part of the information. In this way, a carefully controlled non-harmonious connective can serve as the logical core of a quantum programming language, making the logic of measurement explicit in the proof theory and connecting naturally with semantic models based on completely positive maps [16, 17].

2.3. Linear types and quantum data

Quantum information, however, is subject to stronger constraints than classical information. In particular, the no-cloning and no-deleting theorems show that unknown pure states cannot be copied

or erased by arbitrary operations [11, 12, 22]. Linear logic and linear type systems provide a natural way to reflect these constraints in the structure of proofs and programs [13, 23]. At a high level, a linear assumption is one that must be used exactly once: it cannot be silently duplicated or discarded. By contrast, unrestricted assumptions, often identified by a modality such as ! or by placing them in a separate context, may be used any number of times.

Many existing quantum λ-calculi exploit this distinction by separating quantum and classical data [14, 15, 24]. Quantum data, such as qubits and higher-dimensional systems, live in a linear part of the context and must be consumed linearly, while classical data, including measurement outcomes, reside in an unrestricted part and can be freely copied or thrown away. This separation allows the type system to enforce basic physical principles while still supporting classical control and duplication where appropriate. The present work aims to bring a similar linear discipline to the sup-calculus, so that the logical account of superposition and measurement provided by the non-harmonious connective sup is combined with a type-theoretic guarantee that quantum resources are managed in a physically sound way.

3. A Linear Sup-Calculus: Syntax and Types

We now define the syntax and type system of a linear sup-calculus. The central design choice is to split the typing context into two zones: a zone of *classical* assumptions, which may be weakened and contracted, and a zone of *quantum* assumptions, which must be used exactly once. This separation will allow us to treat quantum data linearly while still permitting free duplication and discarding of classical information.

3.1. Types

We assume a collection of base types and, among them, we distinguish a quantum base type qbit, representing a single qubit, and a classical base type bit, representing a classical bit. On top of these, types are generated by the grammar

$$A, B ::= \mathsf{qbit} \mid \mathsf{bit} \mid 1 \mid A \otimes B \mid A \oplus B \mid A \mathsf{\ sup\ } B \mid A \multimap B.$$

The type 1 is the unit type with a unique inhabitant. The connective $A \otimes B$ is a linear tensor product used to combine resources that must be consumed exactly once. The type $A \oplus B$ represents a classical binary choice, modelling a value that is known to be either of type A or of type B in a way that can be tested by case analysis. The type $A \mathsf{\ sup\ } B$ is the non-harmonious connective introduced earlier and is used here to model quantum superposition and measurement at the level of proofs. Finally, $A \multimap B$ is a linear function type, whose inhabitants consume an argument of type A exactly once in order to produce a result of type B.

We do not introduce an explicit modality such as ! to mark unrestricted types. Instead, we stratify types by their intended usage discipline. Intuitively, types that contain qbit in positive position must be treated as quantum and placed in the linear part of the context, whereas types built solely from bit, 1, and $\oplus$ can safely be considered classical and may live in the unrestricted part. We do not formalise this stratification at the level of the grammar; it is enforced indirectly through the typing rules by controlling how assumptions of each kind may be used.

3.2. Terms

The term language is a linear λ-calculus enriched with algebraic constructs and a dedicated operator for the sup-connective. Variables x, y, z range over term variables, and scalars α, β range over complex numbers. The syntax of terms is given by

$$\begin{aligned} t, u, v \quad ::= \quad & x \mid \star \mid \lambda x{:}A.\, t \mid t\, u \mid (t, u) \mid \mathsf{let}\ (x, y) = t\ \mathsf{in}\ u \\ & \mid \mathsf{inl}\, t \mid \mathsf{inr}\, t \mid \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow u \mid \mathsf{inr}\, y \Rightarrow v \\ & \mid \mathsf{sup}(t, u) \mid \delta_1(t, x.u) \mid \delta_2(t, y.v) \\ & \mid 0_A \mid \alpha \cdot t \mid t + u. \end{aligned}$$

The constant $\star$ is the unique term of type 1. Abstraction $\lambda x{:}A.\, t$ and application $t\, u$ form the usual pair of constructs for functions, but in our setting they are subject to a linear discipline: an argument of type A must be consumed exactly once in the body. The pair constructor (t, u) and the corresponding $\mathsf{let}\ (x, y) = t\ \mathsf{in}\ u$ provide introduction and elimination for the tensor product, allowing us to package and unpack linear resources. The terms $\mathsf{inl}\, t$ and $\mathsf{inr}\, t$ inject into a classical sum type $A \oplus B$, and the case construct performs case analysis, deconstructing such a value and proceeding according to the chosen branch.

The term $\mathsf{sup}(t, u)$ is the introduction form for the connective $A\, \mathsf{sup}\, B$ and intuitively denotes a state in which both t and u are jointly available, as in a quantum superposition. The terms $\delta_1(t, x.u)$ and $\delta_2(t, y.v)$ are the corresponding excessive elimination forms: they take a term t of type $A\, \mathsf{sup}\, B$ and, respectively, project onto the A-branch or the B-branch, binding the chosen component to a variable and discarding the other. These will later be interpreted as measurement operations.

Finally, the constant 0_A is the zero term of type A, and the constructs $\alpha \cdot t$ and $t + u$ equip the term language with scalar multiplication and addition, following the tradition of algebraic λ-calculi. Together, they allow one to form linear combinations of terms, which in the quantum fragment will correspond to superpositions of states or linear combinations of operators. Throughout, we work modulo standard α-equivalence for the renaming of bound variables.

3.3. Typing judgements and structural rules

Typing judgements have the form

$$\Gamma; \Delta \vdash t : A,$$

where Γ is a *classical* context and Δ is a *quantum* context. Both contexts are finite maps from variables to types, and they are distinguished by their structural discipline. Assumptions in Γ behave like ordinary intuitionistic hypotheses: they may be weakened and contracted, so classical variables may be used any number of times, including not at all. Assumptions in Δ behave linearly: each quantum variable must be used exactly once in any well-typed term. We do not write explicit structural rules; instead, this discipline is enforced implicitly through the typing rules by carefully controlling how contexts are split and recombined. When we write Δ_1, Δ_2 we mean the disjoint union of two quantum contexts, identified up to permutation.

3.4. Typing rules

We present the key typing rules for the calculus; further variants, for example for sums constructed from $\oplus$, follow the same pattern.

For variables, the typing rules reflect the two kinds of context. A variable drawn from the classical context Γ may be used without affecting the quantum context, whereas a variable from the quantum context must appear alone in that part of the judgement:

$$\frac{x : A \in \Gamma}{\Gamma; \Delta \vdash x : A}\ \textsc{VarC} \qquad \frac{x : A \in \Delta}{\Gamma; x{:}A \vdash x : A}\ \textsc{VarQ}$$

For the unit type, there is a single closed term:

$$\frac{}{\Gamma; \cdot \vdash \star : 1}\ \textsc{Unit}$$

so $\star$ can be formed without any quantum resources. The rules for the tensor product express that forming a pair requires splitting the available quantum resources between the two components, while eliminating a pair requires using both components linearly:

$$\frac{\Gamma; \Delta_1 \vdash t : A \qquad \Gamma; \Delta_2 \vdash u : B}{\Gamma; \Delta_1, \Delta_2 \vdash (t, u) : A \otimes B}\ \textsc{Tensor-I}$$

$$\frac{\Gamma; \Delta_1 \vdash t : A \otimes B \qquad \Gamma; \Delta_2, x{:}A, y{:}B \vdash u : C}{\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{let}\ (x, y) = t\ \mathsf{in}\ u : C}\ \textsc{Tensor-E}$$

Linear functions are introduced and eliminated in the usual way, but the contexts make explicit that the argument is a linear resource. Introduction abstracts over a quantum variable, and elimination consumes the resources from both the function and its argument:

$$\frac{\Gamma; \Delta, x{:}A \vdash t : B}{\Gamma; \Delta \vdash \lambda x{:}A.\, t : A \multimap B}\ \textsc{Lin-I}$$

$$\frac{\Gamma; \Delta_1 \vdash t : A \multimap B \qquad \Gamma; \Delta_2 \vdash u : A}{\Gamma; \Delta_1, \Delta_2 \vdash t\, u : B}\ \textsc{Lin-E}$$

Classical choice is handled in the standard intuitionistic way, with injections and case analysis. The introduction rules state that a term of type A may be injected on the left into $A \oplus B$, and a term of type B may be injected on the right:

$$\frac{\Gamma; \Delta \vdash t : A}{\Gamma; \Delta \vdash \mathsf{inl}\, t : A \oplus B}\ \textsc{Choice-L} \qquad \frac{\Gamma; \Delta \vdash t : B}{\Gamma; \Delta \vdash \mathsf{inr}\, t : A \oplus B}\ \textsc{Choice-R}$$

The elimination rule for $\oplus$ requires providing a branch for each alternative and uses the same quantum context Δ_2 in both branches, reflecting that the choice itself is classical:

$$\frac{\Gamma; \Delta_1 \vdash t : A \oplus B \qquad \Gamma; \Delta_2, x{:}A \vdash u : C \qquad \Gamma; \Delta_2, y{:}B \vdash v : C}{\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow u \mid \mathsf{inr}\, y \Rightarrow v : C}\ \textsc{Choice-E}$$

For the connective sup we adopt a conjunction-like introduction rule and two excessive elimination rules. The introduction rule states that, to construct a term of type $A\ \mathsf{sup}\ B$, we must separately provide a term of type A and a term of type B, possibly using disjoint parts of the quantum context:

$$\frac{\Gamma; \Delta_1 \vdash t : A \qquad \Gamma; \Delta_2 \vdash u : B}{\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{sup}(t, u) : A\ \mathsf{sup}\ B}\ \textsc{Sup-I}$$

The resulting term expresses simultaneous availability of t and u. The elimination rules, by contrast, each select a single branch and discard the other. In $\delta_1(s, x.t)$, the term s of type $A \,\mathsf{sup}\, B$ is consumed, and the branch proceeds with a fresh variable x of type A:

$$\frac{\Gamma; \Delta_1 \vdash s : A \,\mathsf{sup}\, B \qquad \Gamma; \Delta_2, x{:}A \vdash t : C}{\Gamma; \Delta_1, \Delta_2 \vdash \delta_1(s, x.t) : C}\ \textsc{Sup-E}_1$$

Similarly, $\delta_2(s, y.u)$ projects onto the B-component:

$$\frac{\Gamma; \Delta_1 \vdash s : A \,\mathsf{sup}\, B \qquad \Gamma; \Delta_2, y{:}B \vdash u : C}{\Gamma; \Delta_1, \Delta_2 \vdash \delta_2(s, y.u) : C}\ \textsc{Sup-E}_2$$

Unlike the elimination rule for $\oplus$, there is no requirement to provide both branches at once; each of δ_1 and δ_2 can be used independently. This asymmetry captures the excessiveness of sup: elimination retrieves less structured information than was present at introduction, and in doing so it erases part of the state. Operationally, these rules will correspond to two different measurement operators projecting onto the left and right components.

Finally, we equip each type with an algebraic structure over the complex numbers in line with the algebraic λ-calculus. There is a distinguished zero term 0_A of each type A,

$$\frac{}{\Gamma; \Delta \vdash 0_A : A}\ \textsc{Zero}$$

scalar multiplication by $\alpha \in \mathbb{C}$ preserves types,

$$\frac{\Gamma; \Delta \vdash t : A}{\Gamma; \Delta \vdash \alpha \cdot t : A}\ \textsc{Scalar}$$

and sums of terms are formed by combining their quantum resources:

$$\frac{\Gamma; \Delta_1 \vdash t : A \qquad \Gamma; \Delta_2 \vdash u : A}{\Gamma; \Delta_1, \Delta_2 \vdash t + u : A}\ \textsc{Sum}$$

In the quantum fragment of the calculus these algebraic constructs will model superpositions and linear combinations of states and operators, and the typing discipline ensures that such combinations remain compatible with the linear usage of quantum data.

4. Operational Semantics

We now define an operational semantics for the linear sup-calculus. The dynamics are given by two interacting reduction relations: a deterministic β-reduction relation capturing ordinary computation and algebraic simplification, and a measurement relation induced by the excessive sup-elimination rules, which accounts for quantum measurement steps.

4.1. Values and deterministic reduction

We first specify which terms are considered values, that is, computation results for the deterministic part of the semantics. Values are given by

$$v \;::=\; x \mid \star \mid \lambda x{:}A.\, t \mid (v_1, v_2) \mid \mathsf{inl}\, v \mid \mathsf{inr}\, v \mid \mathsf{sup}(v_1, v_2) \mid 0_A \mid \alpha \cdot v \mid v_1 + v_2.$$

Thus, variables, the unit constant, abstractions, pairs of values, injections of values into sums, sup-terms whose components are themselves values, and algebraic combinations built from 0_A, scalar multiples and sums are all treated as values.

Deterministic reduction, written $\to$, is defined by the usual β-rules together with algebraic rewrites. The β-rule for function application states that applying a linear abstraction to a value results in substitution:

$$(\lambda x{:}A.\, t)\, v \to t[v/x].$$

For tensor pairs, elimination by a let-binding simply unpacks the pair and substitutes its components into the body:

$$\mathsf{let}\ (x, y) = (v_1, v_2)\ \mathsf{in}\ u \to u[v_1/x, v_2/y].$$

Case analysis on a classical sum behaves as expected: if the scrutinee is a left injection, the corresponding branch is selected and the payload substituted,

$$\mathsf{case}\ (\mathsf{inl}\, v)\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow u \mid \mathsf{inr}\, y \Rightarrow w \to u[v/x],$$

and if the scrutinee is a right injection, the right branch is chosen:

$$\mathsf{case}\ (\mathsf{inr}\, v)\ \mathsf{of}\ \mathsf{inl}\, x \Rightarrow u \mid \mathsf{inr}\, y \Rightarrow w \to w[v/y].$$

The algebraic structure is governed by congruence rules that normalise nested scalar products and sums. Multiplication of scalars is associative at the term level, so a scalar applied to another scalar multiple reduces as

$$\alpha \cdot (\beta \cdot v) \to (\alpha\beta) \cdot v,$$

and the unit scalar acts as identity while the zero scalar sends everything to the zero term:

$$1 \cdot v \to v, \qquad 0 \cdot v \to 0_A.$$

Sums are simplified by removing the additive identity and by combining like terms. In particular,

$$t + 0_A \to t, \qquad 0_A + t \to t, \qquad \alpha \cdot v + \beta \cdot v \to (\alpha + \beta) \cdot v.$$

As usual, these rules are closed under evaluation contexts: reduction may occur inside larger terms according to a standard call-by-value or call-by-name strategy (we leave the precise choice implicit here), and we omit the full definition of evaluation contexts for brevity.

4.2. Measurement reduction

The excessive sup-elimination rules describe measurement-like operations on terms of type $A \,\mathsf{sup}\, B$. Intuitively, a term $\mathsf{sup}(v_1, v_2)$ of type $A \,\mathsf{sup}\, B$ represents a quantum superposition with two components, and applying δ_1 or δ_2 corresponds to choosing one of these components by measurement while irreversibly discarding the other. To reflect this, we introduce a separate, non-deterministic reduction relation $t \Rightarrow t'$ devoted to measurement steps.

We abstract away from concrete probabilities and treat measurement as nondeterministic branching. When a sup-term appears directly as the argument of an elimination, we have

$$\delta_1(\mathsf{sup}(v_1, v_2), x.u) \Rightarrow u[v_1/x]$$

and

$$\delta_2(\mathsf{sup}(v_1, v_2), y.u) \Rightarrow u[v_2/y],$$

so δ_1 projects onto the left component of the superposition and δ_2 onto the right component, in both cases substituting the selected value into the continuation and forgetting the other branch. As with deterministic reduction, the relation $\Rightarrow$ is extended contextually so that measurement can occur within larger terms whenever a suitable redex is exposed.

For later use we write $\to^*$ for the reflexive-transitive closure of the deterministic reduction relation and $\Rightarrow^*$ for the reflexive-transitive closure of the measurement relation. The combined operational semantics is then given by the union

$$\rightsquigarrow \; = \; (\to \cup \Rightarrow),$$

again considered up to reflexive-transitive closure when discussing multi-step evaluation. Thus a computation may interleave deterministic computation steps and measurement steps in any order allowed by the reduction rules.

4.3. Subject reduction

The key meta-theoretic property of this operational semantics is that it preserves typing. In other words, evaluation never produces an ill-typed term, and the linear discipline on quantum resources is maintained throughout reduction.

Theorem 4.1 (Subject reduction)**.** *If* $\Gamma; \Delta \vdash t : A$ *and* $t \rightsquigarrow t'$*, then* $\Gamma; \Delta \vdash t' : A$*.*

Proof. The proof proceeds by induction on the derivation of $t \rightsquigarrow t'$, distinguishing deterministic and measurement steps. For deterministic steps, the argument is standard: one verifies case by case that each β-reduction and each algebraic rewriting preserves typing. This relies on substitution lemmas adapted to the two-context setting, ensuring that replacing a variable by a term of the same type in either the classical or the quantum context yields another well-typed term with the same overall judgement.

For measurement steps, consider for instance a reduction of the form

$$\delta_1(\mathsf{sup}(v_1, v_2), x.u) \Rightarrow u[v_1/x].$$

By the typing rules, a derivation of the premise must provide a typing for the sup-term and for the continuation. Thus we know that there is a context Δ_1 such that

$$\Gamma; \Delta_1 \vdash \mathsf{sup}(v_1, v_2) : A \,\mathsf{sup}\, B,$$

and another context Δ_2 such that

$$\Gamma; \Delta_2, x{:}A \vdash u : C.$$

From these two judgements, the rule for sup-elimination (SUP-E$_1$) yields a typing

$$\Gamma; \Delta_1, \Delta_2 \vdash \delta_1(\mathsf{sup}(v_1, v_2), x.u) : C.$$

By sup-introduction, the typing of $\mathsf{sup}(v_1, v_2)$ in turn arises from typings

$$\Gamma; \Delta_{11} \vdash v_1 : A \qquad \text{and} \qquad \Gamma; \Delta_{12} \vdash v_2 : B$$

with Δ_1 equal to the union Δ_{11}, Δ_{12}. A substitution lemma for the quantum context then guarantees that replacing the variable x of type A by the value v_1 of the same type in the derivation of u yields a term $u[v_1/x]$ that is well-typed under the combined context $\Gamma; \Delta_{11}, \Delta_2 \vdash u[v_1/x] : C$. Since Δ_{11}, Δ_2 together with Δ_{12} reconstruct the original quantum context, this shows that the conclusion of the measurement step is well-typed under the same overall context as the premise. The case for δ_2 is entirely analogous.

The remaining reduction rules are treated in a similar way, either by straightforward use of the substitution lemmas or by simple inspection of the typing rules that generated the redex. This completes the sketch of the argument. □

As an immediate corollary, the type discipline guarantees that reduction does not produce ill-typed terms, and in particular that linear quantum resources are neither leaked nor duplicated by the operational rules. Every evaluation sequence starting from a well-typed term preserves the separation between classical and quantum contexts, ensuring that the operational semantics is consistent with the intended resource-sensitive interpretation of types.

5. Linearity and No-Cloning

In this section we analyse in more detail the linearity properties enforced by the type system and explain how they lead to a no-cloning result for quantum data. The central idea is to isolate a purely quantum fragment of the calculus, show that all well-typed terms in this fragment denote linear maps, and then use this fact together with the standard no-cloning theorem from quantum mechanics to rule out the existence of a generic duplicator for qubits [11, 12, 21]. Our treatment is in the spirit of algebraic and linear λ-calculi for quantum computation [9, 10, 14, 15], but specialises to the setting of the sup-calculus.

5.1. Quantum fragment and linearity

We first define the *quantum fragment* of the calculus. This is the part of the language where both types and contexts are purely quantum. Formally, we consider those well-typed terms whose result types are built from qbit and 1 using only the connectives $\otimes$, sup, and $\multimap$, and whose typing judgements have the form

$$\cdot; \Delta \vdash t : A,$$

that is, there are no classical assumptions at all and all variables appear in the linear, quantum context. In this fragment there is no duplication or discarding of hypotheses by construction, and all connectives are interpreted as quantum constructors, much as in other linear quantum λ-calculi [14, 24, 25].

The following proposition states that, when restricted to this fragment, every constructor of the language behaves linearly with respect to the algebraic structure of scalars and sums introduced earlier [9, 10].

Proposition 5.1 (Linearity of constructors)**.** *All term formers of the quantum fragment are linear with respect to the algebraic structure. In particular, abstraction and application are linear in their arguments, pairing and* sup*-introduction are bilinear in their components, and all elimination forms are linear in their major argument and in each branch they contain.*

Proof. The proof proceeds by structural induction on typing derivations restricted to the quantum fragment, following the usual pattern for algebraic λ-calculi [9, 10]. Each typing rule in this fragment combines contexts using disjoint unions of quantum resources and introduces a term using one of the connectives $\otimes$, sup, or $\multimap$, or using the scalar and sum constructors. There is no rule that copies or discards assumptions in Δ, and there is no rule that introduces non-linear behaviour into the term language.

When such a system is interpreted over a vector-space semantics, each type in the quantum fragment is associated with a Hilbert space, and each typing rule corresponds to a linear construction on these spaces: tensor products, function spaces realised as spaces of linear maps, direct sums for sup, and linear combinations given by scalars and addition [26, 27]. Because the typing rules only compose these linear constructions in a way that respects the algebraic structure, the denotation of every constructor is a linear map in each of its arguments. This yields linearity of abstraction, application, pairing, sup-introduction and the various elimination forms, as claimed. □

Thus, in the quantum fragment the calculus behaves like an algebraic linear λ-calculus enriched with the sup-connective: all well-typed terms denote linear operators between the Hilbert spaces associated with their types [16, 17].

5.2. A no-cloning theorem in the calculus

We now show how these linearity properties lead to a no-cloning result. Intuitively, one would like to ask whether the calculus can define a term that, given an arbitrary qubit as input, produces two identical copies of it. A term of type $\mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$ would play exactly this role: it would consume one qubit and return a pair of qubits that, operationally, should each be in the same state as the input.

The next theorem states that such a term cannot exist in the purely quantum fragment, echoing the no-cloning theorem in quantum information theory [11, 12, 28].

Theorem 5.2 (No cloning)**.** *There is no closed term t such that*

$$\cdot;\cdot \vdash t : \mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$$

whose free variables and internal definitions lie entirely in the quantum fragment.

Proof. Suppose, for the sake of contradiction, that such a term t exists. By Proposition 5.1, and more generally by the linearity of the quantum fragment, the denotation of t in a Hilbert-space semantics must be a linear map

$$[\![t]\!] : \mathbb{C}^2 \to \mathbb{C}^2 \otimes \mathbb{C}^2,$$

where $\mathbb{C}^2$ is the Hilbert space associated with a single qubit. Moreover, by type soundness and by the intended semantics of the connectives, this map is definable by composing tensor products, linear abstractions and applications, sup-constructors and eliminations (interpreted as measurements), and algebraic linear combinations. All of these constructions are known to yield completely positive maps arising from unitary evolution and projective measurements; in particular, they are linear in the input state [21, 29, 30].

If t were to act as a universal duplicator for qubits, then for every unit vector $|\psi\rangle$ in $\mathbb{C}^2$ we would have

$$[\![t]\!](|\psi\rangle) \simeq |\psi\rangle \otimes |\psi\rangle$$

(up to the usual identification of pure states with one-dimensional density operators). However, the standard no-cloning theorem shows that there is no unitary operator, and in fact no completely positive trace-preserving map, that implements this behaviour for all pure states [11, 12, 28]. The contradiction arises from the linearity condition: on the one hand, linearity forces

$$[\![t]\!](\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,[\![t]\!](|0\rangle) + \beta\,[\![t]\!](|1\rangle),$$

while cloning would require

$$[\![t]\!](\alpha\,|0\rangle + \beta\,|1\rangle) = (\alpha\,|0\rangle + \beta\,|1\rangle)^{\otimes 2},$$

and these two expressions cannot agree for arbitrary complex α and β unless $[\![t]\!]$ is trivial. Since every closed term in the quantum fragment must denote a linear (indeed completely positive) map, and no such map can satisfy the cloning equations, we conclude that no term of the required type with the intended duplicating behaviour can exist. A more formal argument would proceed by proving an adequacy result relating the operational semantics of t to its denotation and then invoking the standard no-cloning theorem in the semantic model [16, 17]. □

The absence of a term of type $\mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$ in the quantum fragment thus mirrors the physical impossibility of cloning unknown quantum states [21, 27].

Remark 5.3. The no-cloning theorem established above is specific to quantum data and does not preclude duplication of classical information. For instance, a term of type $\mathsf{bit} \Rightarrow \mathsf{bit} \otimes \mathsf{bit}$ may be definable when the input bit is taken from the classical context Γ. In that setting, weakening and contraction allow the same classical variable to be used twice, so the type system permits a function that takes a classical bit and returns a pair of identical copies. The calculus therefore distinguishes sharply between quantum and classical duplication: linearity forbids generic cloning of qubits, while unrestricted structural rules in the classical context allow duplication of classical outcomes, such as measurement results [13, 23].

6. Denotational Semantics

We briefly sketch a denotational semantics for the linear sup-calculus that justifies the informal semantic reading used above. The aim is to associate each type with a finite-dimensional Hilbert space and each well-typed term with a completely positive map between spaces of density operators, in such a way that the sup-connective is interpreted as a direct sum and its elimination forms correspond to quantum measurements [17, 21, 29, 30].

6.1. Types as Hilbert spaces

Each type is interpreted as a finite-dimensional Hilbert space, or more precisely as the carrier of a suitable C*-algebra [27, 31]. For the basic types we set

$$\begin{aligned} [\![\mathsf{qbit}]\!] &= \mathbb{C}^2, \\ [\![\mathsf{bit}]\!] &= \mathbb{C}^2 \text{ with a distinguished orthonormal basis } \{0,1\}, \\ [\![1]\!] &= \mathbb{C}, \end{aligned}$$

so that both quantum and classical bits are modelled as two-dimensional spaces, but classical bits come equipped with a fixed computational basis. Composite types are interpreted using standard constructions:

$$\begin{aligned}
[\![A \otimes B]\!] &= [\![A]\!] \otimes [\![B]\!], \\
[\![A \oplus B]\!] &= [\![A]\!] \oplus [\![B]\!], \\
[\![A \;\mathsf{sup}\; B]\!] &= [\![A]\!] \oplus [\![B]\!], \\
[\![A \multimap B]\!] &= \mathcal{L}([\![A]\!], [\![B]\!]),
\end{aligned}$$

where $\otimes$ denotes the Hilbert-space tensor product, $\oplus$ denotes the direct sum, and $\mathcal{L}(H, K)$ is the space of linear operators from H to K [27, 32]. At the level of carriers, $\oplus$ and sup are thus both interpreted as direct sums; the difference between them is reflected not in their underlying spaces but in the operational behaviour of their introduction and elimination forms, and hence in the way terms of these types are interpreted as maps between density operators.

6.2. Terms as completely positive maps

To interpret terms, we move from pure vectors to density operators. Given a Hilbert space H, let $\mathcal{D}(H)$ denote the convex set of density operators on H [21, 29]. A typing judgement

$$\Gamma; \Delta \vdash t : A$$

is interpreted as a completely positive map between tensor products of such spaces. For simplicity, suppose that all types appearing in the quantum context Δ are genuine quantum types and that all types in the classical context Γ are classical. Then the input system associated with Δ is

$$[\![\Gamma; \Delta]\!] = \bigotimes_{x:A \in \Delta} \mathcal{D}([\![A]\!]),$$

where the tensor product is taken over all quantum variables in the context. A closed term $\cdot; \cdot \vdash t : A$ has no free variables and is therefore interpreted as a completely positive map

$$[\![t]\!] : \mathcal{D}(\mathbb{C}) \to \mathcal{D}([\![A]\!]) \cong \mathcal{D}([\![A]\!]),$$

which we can identify with a CP map on $\mathcal{D}([\![A]\!])$ by treating the scalar input as a trivial system [17, 30].

The sup-constructors receive a specific interpretation in this framework.

Definition 6.1 (Interpretation of sup)**.** Let t and u be terms that, under a common environment, denote completely positive maps E_1 and E_2 with codomains $\mathcal{D}([\![A]\!])$ and $\mathcal{D}([\![B]\!])$ respectively. The introduction form $\mathsf{sup}(t, u)$ is interpreted as the block-diagonal CP map

$$\rho \mapsto E_1(\rho) \oplus E_2(\rho)$$

on $\mathcal{D}([\![A]\!] \oplus [\![B]\!])$, which prepares a state in the direct sum by placing the outputs of E_1 and E_2 in the two summands. The elimination forms δ_1 and δ_2 are interpreted as the two projective measurements associated with the canonical projections π_1 and π_2 of the direct sum. Concretely, they correspond to CP instruments that first perform the projection onto the A or B summand and then apply the interpretation of the continuation $x.u$ or $y.v$ as appropriate, in the sense of quantum measurement theory [33–35].

A fully formal treatment requires careful bookkeeping of how classical and quantum data are combined, as well as explicit modelling of probabilistic branching produced by measurements. The crucial point, however, is that each typing rule induces a well-defined CP construction, and that the combination of sup-introduction with its excessive elimination rules yields CP instruments of the kind used to model quantum measurements in quantum information theory [17, 21, 27].

Theorem 6.2 (Adequacy, informal)**.** *For each well-typed closed term*

$$\cdot;\cdot \vdash t : A$$

in the quantum fragment there exists a completely positive map

$$[\![t]\!] : \mathcal{D}([\![A]\!]) \to \mathcal{D}([\![A]\!])$$

that is consistent with the operational semantics. In particular, if $t \rightsquigarrow^* v$ by a sequence of deterministic and measurement reductions, then the action of $[\![t]\!]$ on density operators agrees with the effect of this reduction sequence, up to the expected identification of non-deterministic branching in the operational semantics with different measurement outcomes in the denotational model [16, 36, 37].

A precise adequacy theorem would require working in a suitable categorical setting of completely positive maps, formulating the operational semantics as a labelled transition system on density operators, and proving that each reduction step corresponds to an equality or refinement of denotational interpretations [17, 27, 30]. Developing this in full generality lies beyond the scope of the present exposition; here the informal statement serves to justify the use of the Hilbert-space model in the linearity and no-cloning arguments above.

7. Examples

We illustrate the linear sup-calculus by looking at small quantum programs and how their types and terms reflect the intended behaviour. We begin with the Hadamard gate on a single qubit and a simple measurement in the computational basis, and then consider two slightly richer examples: the construction of a Bell state and a program that applies a unitary operation controlled by a measurement outcome.

7.1. The Hadamard gate

The Hadamard gate H is a standard unitary operator on qbit defined on the computational basis by

$$H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad H\,|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

In our setting, we regard $|0\rangle$ and $|1\rangle$ as two distinguished closed terms of type qbit, and we represent the Hadamard gate as a term

$$H : \mathsf{qbit} \multimap \mathsf{qbit}.$$

At the level of equations, its behaviour on the basis states can be given schematically as

$$H\ |0\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle\,, \qquad H\ |1\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle\,.$$

The algebraic structure of the calculus then determines its action on arbitrary superpositions by linearity. For example, if a qubit is in the state

$$\alpha \cdot |0\rangle + \beta \cdot |1\rangle ,$$

then the term H satisfies

$$H(\alpha \cdot |0\rangle + \beta \cdot |1\rangle) \equiv \alpha \cdot H\,|0\rangle + \beta \cdot H\,|1\rangle ,$$

and substituting the defining equations for $H\,|0\rangle$ and $H\,|1\rangle$ recovers the usual matrix action of the Hadamard gate. The linear type $\mathsf{qbit} \multimap \mathsf{qbit}$ ensures that the input qubit is consumed exactly once; there is no way to write a term of this type that inspects a qubit twice or duplicates it in the process of applying H.

Syntactically, one may encode H as a term that pattern-matches on abstract constructors representing $|0\rangle$ and $|1\rangle$ and returns the corresponding linear combination, for instance in a style where the two basis states are represented by distinct canonical values and the algebraic operations $\alpha \cdot t$ and $t + u$ are used to assemble the result.

7.2. Measurement in the computational basis

As a second example, we define a measurement operation in the computational basis Z,

$$\mathsf{meas}_Z : \mathsf{qbit} \multimap \mathsf{bit} \otimes \mathsf{qbit}.$$

Intuitively, meas_Z takes an input qubit, measures it in the $\{|0\rangle , |1\rangle\}$ basis, returns the classical outcome as a bit, and also returns the post-measurement qubit. The type expresses that the overall map is linear in its quantum input and that the result consists of a classical part (the bit) and a quantum part (the collapsed qubit).

At a high level, the implementation of meas_Z in the linear sup-calculus proceeds in two steps. First, one views the input qubit as inhabiting a type of the form $\mathsf{qbit}\ \mathsf{sup}\ \mathsf{qbit}$, where the two components of sup correspond to the two basis branches. This reflects the idea that, prior to measurement, the system behaves like a superposition with two labelled outcomes. Second, one uses the elimination forms δ_1 and δ_2 to project onto one branch or the other:

$$\delta_1(s, x.u) \qquad \text{and} \qquad \delta_2(s, y.v),$$

where s has type $A\,\mathsf{sup}\,B$ and the continuations u and v construct appropriate outputs for each case. In the present situation, A and B are both instances of qbit, and the continuations produce a tensor of a classical bit and a qubit, such as $(0, x)$ in the first branch and $(1, y)$ in the second.

Concretely, if we assume constructors $\mathsf{b0}$ and $\mathsf{b1}$ of type bit for the classical outcomes, we may think of meas_Z as a term that, when applied to a prepared superposition, reduces by measurement steps of the form

$$\mathsf{meas}_Z\ |0\rangle \rightsquigarrow (\mathsf{b0}, |0\rangle), \qquad \mathsf{meas}_Z\ |1\rangle \rightsquigarrow (\mathsf{b1}, |1\rangle),$$

with the understanding that if the input is an arbitrary superposition then the operational semantics branches non-deterministically according to the two outcomes. The typing judgement for such a term has the form

$$\Gamma, r{:}\mathsf{bit}\ ;\ q{:}\mathsf{qbit} \vdash \mathsf{meas}_Z\, q : \mathsf{bit} \otimes \mathsf{qbit},$$

where the bit r is placed in the classical context Γ and can therefore be duplicated or forgotten, while the qubit q remains in the linear context and must be used exactly once in subsequent computation. This illustrates how measurement turns quantum information into classical control without violating the linear usage discipline for the remaining quantum state.

7.3. Creating a Bell state

A slightly more involved example is the construction of an entangled Bell state from two qubits. Consider the usual Bell state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$

which can be prepared by starting from the classical state $|00\rangle$, applying a Hadamard gate H to the first qubit, and then applying a controlled-NOT gate (CNot) from the first to the second.

In the linear sup-calculus we may represent this preparation as a term

$$\mathsf{bell} : \mathsf{qbit} \otimes \mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}.$$

Assuming we have terms $H : \mathsf{qbit} \multimap \mathsf{qbit}$ and $\mathsf{CNot} : \mathsf{qbit} \otimes \mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$, a schematic definition is

$$\mathsf{bell} = \lambda p{:}\mathsf{qbit} \otimes \mathsf{qbit}.\ \mathsf{let}\ (q_1, q_2) = p\ \mathsf{in}\ \mathsf{CNot}(H\, q_1, q_2).$$

Typing this term uses the tensor elimination rule to unpack the input pair (q_1, q_2), the linear function type for H and CNot, and the tensor introduction rule to form the output pair. Because the context is purely quantum, the two input qubits are consumed exactly once in building the Bell state. Applying bell to the canonical pair $(|0\rangle, |0\rangle)$ yields, up to algebraic equalities, a term whose denotation is $|\Phi^+\rangle$.

This example shows how familiar circuit constructions are naturally expressed in the calculus: tensor types model multiple qubits, linear function types model gates, and the algebraic structure captures superposition and interference.

7.4. Classical control after measurement

As a final example, we consider a program that measures a qubit and then applies a further unitary gate depending on the classical result. This pattern of *classical control* is ubiquitous in quantum algorithms and provides a good test of the interaction between the classical and quantum contexts.

Suppose U_0 and U_1 are two unitary operators on qbit, represented as terms

$$U_0, U_1 : \mathsf{qbit} \multimap \mathsf{qbit}.$$

We define a controlled operation

$$\mathsf{ctrl} : \mathsf{qbit} \multimap \mathsf{qbit}$$

that measures its input and then applies U_0 in case of outcome 0 and U_1 in case of outcome 1. Informally, the behaviour is

$$\mathsf{ctrl}\, q \rightsquigarrow \begin{cases} U_0\, q' & \text{if measurement of } q \text{ yields 0 and leaves } q', \\ U_1\, q' & \text{if measurement of } q \text{ yields 1 and leaves } q'. \end{cases}$$

In the calculus, ctrl can be built by composing meas_Z with a case analysis on the classical outcome. A schematic term is

$$\mathsf{ctrl} = \lambda q{:}\mathsf{qbit}.\ \mathsf{let}\ (b, q') = \mathsf{meas}_Z\, q\ \mathsf{in}\ \mathsf{case}\ b\ \mathsf{of}\ \mathsf{b0} \Rightarrow U_0\, q' \mid \mathsf{b1} \Rightarrow U_1\, q'.$$

Typing this term uses the fact that meas_Z has type $\mathsf{qbit} \multimap \mathsf{bit} \otimes \mathsf{qbit}$. The let binding unpacks the result into a classical bit b and a qubit q'. The bit b is placed in the classical context and may be duplicated by the case construct if needed, while the qubit q' remains in the quantum context and is passed linearly to either U_0 or U_1. The type of ctrl is still $\mathsf{qbit} \multimap \mathsf{qbit}$, expressing that overall the program consumes a qubit once and returns a qubit, but internally it branches according to a classical control structure that arises from measurement.

This example highlights the intended separation of roles: measurement uses the excessive sup-elimination rules to convert quantum information into a classical outcome; the classical result is then manipulated using unrestricted structural rules, while the remaining quantum data continues to be governed by linearity. In combination, the examples in this section show that the linear sup-calculus is expressive enough to encode basic quantum gates, measurements, entanglement, and classical control, all while respecting the linear usage of quantum resources enforced by the type system.

8. Related Work

The present work builds most directly on the sup-logic and sup-calculus introduced by Díaz-Caro and Dowek [7, 8], in which a non-harmonious connective equipped with both harmonious and excessive rules is used to give a proof-theoretic account of quantum measurement. In their setting, the connective $A \,\mathsf{sup}\, B$ has conjunction-like introduction rules and a mixture of disjunction-like and excessive elimination rules, and the associated term calculus can be extended with scalars and finite sums to serve as a core language for quantum computation [9, 10]. Our contribution can be seen as a refinement of that proposal: we keep the same logical insight that non-harmonious elimination captures information loss due to measurement, but we impose a linear type discipline that ensures well-typed quantum programs denote linear, physically meaningful maps and that a no-cloning principle is enforced at the type level rather than only at the semantic level [11, 12].

Beyond the specific case of sup, there is a substantial body of research on quantum λ-calculi and linear type systems for quantum programming. Algebraic λ-calculi with scalars, such as those studied by Arrighi and Díaz-Caro and by Vaux [9, 10], introduce linear combinations of terms and equip the term language with a vector-space structure, much like what we adopt here for the quantum fragment. Other systems, including the calculi of Selinger and Valiron and of Altenkirch and Grattage [14, 15, 24], combine linear types with explicit quantum data types and constructs for unitary evolution and measurement, often supported by categorical models based on symmetric monoidal or dagger-compact categories and on the CPM construction for completely positive maps [16, 27, 30]. Our linear sup-calculus can be viewed as inserting the sup-connective, with its explicitly non-harmonious elimination rules, into this landscape: it inherits the resource-sensitive discipline of linear quantum λ-calculi while providing a proof-theoretic explanation of measurement rooted in the analysis of introduction and elimination rules [1, 2, 5].

The categorical perspective on quantum computation, as developed for example in Abramsky and Coecke's categorical quantum mechanics [17, 26], also informs our approach. There, quantum protocols are modelled in compact closed categories with biproducts, and completely positive maps arise via suitable constructions on these categories, such as the CPM and CP* constructions [27, 30]. In our setting, types are interpreted as finite-dimensional Hilbert spaces and terms as completely positive maps on density operators, and the sup-connective is interpreted as a direct sum whose elimination rules correspond to measurement instruments. While we do not fully develop a categor-

ical semantics in this paper, the informal Hilbert-space model we sketch is compatible with these categorical accounts and suggests that the linear sup-calculus could be situated within a monoidal category of CP maps equipped with a sum-like structure for sup [35, 36].

Finally, our system is closely related to linear logic and its many variants, which have long served as a logical foundation for resource-sensitive computation. Girard's linear logic distinguishes between linear resources, which must be used exactly once, and unrestricted resources, which may be copied and discarded[13]; this distinction has been widely exploited in logical embeddings of quantum computation and in the design of quantum type systems [23, 25]. The two-context discipline we adopt, separating classical and quantum assumptions, echoes the standard separation between an unrestricted context and a linear context in intuitionistic linear logic, and the way we treat measurement outcomes as classical data aligns with approaches that model classical control over quantum data [14, 21]. In this sense, the linear sup-calculus can be viewed as combining three strands: the proof-theoretic study of harmony and excessiveness [3, 19], the linear-logical treatment of resources, and the semantic models of quantum computation based on Hilbert spaces and completely positive maps [16, 17, 27].

9. Conclusion

We have introduced a linear sup-calculus that combines the non-harmonious connective of Díaz-Caro and Dowek with a resource-sensitive type discipline separating quantum from classical data. Typing judgements are stratified into a classical context, where weakening and contraction are allowed, and a quantum context, where assumptions must be used exactly once. Within this framework we defined the syntax, typing rules, and operational semantics of the calculus, distinguishing deterministic β-reduction from measurement steps induced by the excessive elimination rules for sup. We proved subject reduction for the combined reduction relation, ensuring that evaluation preserves types and that the linear usage of quantum resources is maintained. By isolating a purely quantum fragment, we showed that all constructors are linear with respect to the algebraic structure and used this to derive a no-cloning result: there is no closed term of type $\mathsf{qbit} \multimap \mathsf{qbit} \otimes \mathsf{qbit}$, even though classical duplication remains available in the unrestricted context.

To support this interpretation, we sketched a denotational semantics in which types are interpreted as finite-dimensional Hilbert spaces and terms as completely positive maps on density operators, with sup realised as a direct sum and its elimination rules corresponding to measurement instruments. The examples we discussed—such as the Hadamard gate, measurement in the computational basis, preparation of a Bell state, and classically controlled unitaries—illustrate that the calculus can express standard quantum constructions while respecting linearity. These results position the linear sup-calculus at the intersection of proof theory, linear logic, and quantum programming languages. Future work includes developing a full categorical semantics with an adequacy theorem, exploring sequent-calculus formulations with cut-elimination, and extending the calculus with richer type formers to support more sophisticated reasoning about quantum programs and measurements.

References

[1] Gentzen, G. (1964). Investigations into logical deduction. American Philosophical Quarterly, 1(4), 288-306.

[2] Prawitz, D. (2006). *Natural Deduction: A Proof-Theoretical Study.* Courier Dover Publications.

[3] Dummett, M. (1991). *The Logical Basis of Metaphysics.* Harvard university press.

[4] Prior, A. N. (1960). The runabout inference-ticket. *Analysis, 21*(2), 38-39.

[5] Schroeder-Heister, P. (2012). *Proof-Theoretic Semantics.*

[6] Longo, G., & Mossio, M. (2020). Geocentrism vs genocentrism: theories without metaphors, metaphors without theories. *Interdisciplinary Science Reviews, 45*(3), 380-405.

[7] Brunet, C., de Madron, X. D., Guieu, C., Sempéré, R., Conan, P., Cossa, D., ... & Verney, R. (2011). Marine ecosystems' responses to climatic and anthropogenic forcings in the Mediterranean. *Progress in Oceanography, 91*(2), 97-166.

[8] Díaz-Caro, A., & Dowek, G. (2023). A new connective in natural deduction, and its application to quantum computing. *Theoretical Computer Science, 957,* 113840.

[9] Arrighi, P., & Díaz-Caro, A. (2012). A System F accounting for scalars. *Logical Methods in Computer Science, 8*, 3.

[10] Vaux, L. (2009). The algebraic lambda calculus. *Mathematical Structures in Computer Science, 19*(5), 1029-1059.

[11] Wootters, W. K., & Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature, 299*(5886), 802-803.

[12] Dieks, D. G. B. J. (1982). Communication by EPR devices. *Physics Letters A, 92*(6), 271-272.

[13] Girard, J. Y. (1987). Linear logic. *Theoretical Computer Science, 50*(1), 1-101.

[14] Selinger, P., & Valiron, B. (2006). A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science, 16*(3), 527-552.

[15] Altenkirch, T., & Grattage, J. (2005, June). A functional quantum programming language. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)* (pp. 249-258). IEEE.

[16] Selinger, P. (2004). Towards a quantum programming language. *Mathematical Structures in Computer Science, 14*(4), 527-586.

[17] Abramsky, S., & Coecke, B. (2009). Categorical quantum mechanics. *Handbook of Quantum Logic and Quantum Structures, 2*, 261-325.

[18] Tennant, N. (1997). *The Taming of the True.* Oxford University Press.

[19] Read, S. (2000). Harmony and autonomy in classical logic. *Journal of Philosophical Logic, 29*(2), 123-154.

[20] Murzi, J. (2020). Classical harmony and separability. *Erkenntnis, 85*(2), 391-415.

[21] Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information.* Cambridge university press.

[22] Kumar Pati, A., & Braunstein, S. L. (2000). Impossibility of deleting an unknown quantum state. *Nature, 404*(6774), 164-165.

[23] Wadler, P. (1990, April). Linear types can change the world!. In *Programming Concepts and Methods* (Vol. 3, No. 4, p. 5).

[24] Van Tonder, A. (2004). A lambda calculus for quantum computation. *SIAM Journal on Computing, 33*(5), 1109-1135.

[25] Benton, P. N. (1994, September). A mixed linear and non-linear logic: Proofs, terms and models. In *International Workshop on Computer Science Logic* (pp. 121-135). Berlin, Heidelberg: Springer Berlin Heidelberg.

[26] Abramsky, S. (1993). Computational interpretations of linear logic. *Theoretical Computer Science, 111*(1-2), 3-57.

[27] Heunen, C., & Vicary, J. (2019). *Categories for Quantum Theory: An Introduction*. Oxford University Press.

[28] Barnum, H., Caves, C. M., Fuchs, C. A., Jozsa, R., & Schumacher, B. (1996). Noncommuting mixed states cannot be broadcast. *Physical Review Letters, 76*(15), 2818.

[29] Kraus, K., Böhm, A., Dollard, J. D., & Wootters, W. H. (Eds.). (1983). *States, Effects, and Operations Fundamental Notions of Quantum Theory: Lectures in Mathematical Physics at the University of Texas at Austin*. Berlin, Heidelberg: Springer Berlin Heidelberg.

[30] Selinger, P. (2007). Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical Computer Science, 170*, 139-163.

[31] Takesaki, M. (2003). *Theory of Operator Algebras II* (Vol. 125, p. 518). Berlin: Springer.

[32] Reed, M., & Simon, B. (1980). *Methods of Modern Mathematical Physics: Functional Analysis* (Vol. 1). Gulf Professional Publishing.

[33] Davies, E. B., & Lewis, J. T. (1970). An operational approach to quantum probability. *Communications in Mathematical Physics, 17*(3), 239-260.

[34] Ozawa, M. (1984). Quantum measuring processes of continuous observables. *Journal of Mathematical Physics, 25*(1), 79-87.

[35] Chiribella, G., Toigo, A., & Umanità, V. (2013). Normal completely positive maps on the space of quantum operations. *Open Systems & Information Dynamics, 20*(01), 1350003.

[36] Jacobs, B. (2015). New directions in categorical logic, for classical, probabilistic and quantum logic. *Logical Methods in Computer Science, 11*(3),1–76.

[37] Stein, D., & Staton, S. (2021, June). Compositional semantics for probabilistic programs with exact conditioning. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (pp. 1-13). IEEE.

How to cite this article: Gilles Dowek (2024). A Linear Sup-Calculus for Quantum Measurement. *Bulletin of Computer and Data Sciences*, 5(4), 12-32. DOI: 10.71448/bcds2454-2

Received: 24/06/2024 **Revised:** 13/10/2024 **Accepted:** 01/11/2024 **Publish:** 30/12/2024

Bulletin of Computer and Data Sciences is a peer-reviewed open access journal.