Algebraic Reasoning of Quantum Programs via Non-idempotent Kleene Algebra

Yuxiang Peng
University of Maryland, USA

Mingsheng Ying
Chinese Academy of Sciences, China
Tsinghua University, China

Xiaodi Wu
University of Maryland, USA

Abstract

We investigate the *algebraic* reasoning of quantum programs inspired by the success of classical program analysis based on Kleene algebra. One prominent example of such is the famous Kleene Algebra with Tests (KAT), which has furnished both theoretical insights and practical tools. The succinctness of algebraic reasoning would be especially desirable for scalable analysis of quantum programs, given the involvement of exponential-size matrices in most of the existing methods. A few key features of KAT including the idempotent law and the nice properties of classical tests, however, fail to hold in the context of quantum programs due to their unique quantum features, especially in branching. We propose Non-idempotent Kleene Algebra (NKA) as a natural alternative and identify complete and sound semantic models for NKA as well as their quantum interpretations. In light of applications of KAT, we demonstrate algebraic proofs in NKA of quantum compiler optimization and the normal form of quantum **while**-programs. Moreover, we extend NKA with Tests (i.e., NKAT), where tests model quantum predicates following effect algebra, and illustrate how to encode propositional quantum Hoare logic as NKAT theorems.

CCS Concepts: • **Theory of computation** → **Algebraic language theory**; *Equational logic and rewriting*.

Keywords: non-idempotent Kleene algebra, compiler optimization, normal form theorem, quantum Hoare logic.

ACM Reference Format:
Yuxiang Peng, Mingsheng Ying, and Xiaodi Wu. 2022. Algebraic Reasoning of Quantum Programs via Non-idempotent Kleene Algebra. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI '22), June 13–17, 2022, San Diego, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519939.3523713

ACM ISBN 978-1-4503-9265-5/22/06…$15.00
https://doi.org/10.1145/3519939.3523713

1 Introduction

1.1 Background and Motivation

Kleene algebra (KA) [31] that establishes the equivalence of regular expressions and finite automata is an important connection built between programming languages and abstract machines with a wide range of applications. One very successful extension of KA, called Kleene algebra with tests (KAT), was introduced by Kozen [33] that combines KA with Boolean algebra (BA) to model the fundamental constructs arising in programs: sequencing, branching, iteration, etc. More importantly, the equational theory of KAT, which can be finitely axiomatized [37], allows *algebraic reasoning* about corresponding classical programs.

The mathematical elegance and succinctness of algebraic reasoning with KAT have furnished deep theoretical insights as well as practical tools. A lot of topics can be investigated with KAT including, e.g., program transformations [4], compiler optimization [36], Hoare logic [34], and so on. An important recent application of KAT is NetKAT [3] that reasons about the packet-forwarding behavior of software-defined networks, with both a solid theoretical foundation [21] and scalable practical performance [3]. An efficient fragment of KAT, called Guarded KAT (GKAT), has also been identified [55] to model typical imperative programs with an almost linear time equational theory. In contrast, KAT's equational theory is **PSPACE**-complete [14].

Quantum computation has been a topic of significant recent interest. With breakthroughs in experimental quantum computing and the introduction of many quantum programming languages such as Quipper [26], Scaffold [1], QWIRE [46], Microsoft's Q# [58], IBM's Qiskit [2], Google's Cirq [24], Rigetti's Forest [49], there is an imperative need for the analysis and verification of quantum programs.

Indeed, program analysis and verification have been a central topic ever since the seminal work on quantum programming languages [25, 45, 50, 51, 53]. There have been many attempts of developing Hoare-like logic [28] for verification of quantum programs [5, 10, 12, 18, 29, 63]. In particular, D'Hondt and Panangaden [15] proposed the notions of quantum predicate and weakest precondition. Ying [63] established the quantum Hoare logic with (relative) completeness for reasoning about a quantum extension of the **while**-language with many subsequent developments [41, 66, 69]. We refer curious readers to surveys [23, 52, 65] for details.

Quantum **while**-programs have similar (yet semantically different) fundamental constructs (e.g., sequencing, branching, iterations) like classical ones, which gives rise to a natural question of the possibility of using KA/KAT to algebraically reason about quantum programs. Existing methods for quantum program analysis and verification usually involve exponential-size matrices in terms of the system size, which hence significantly limits the scalability. In contrast, a succinct KA-based algebraic reasoning, if possible, would greatly increase the scalability of such analyses for quantum programs due to its mathematical succinctness.

1.2 Research Challenges and Solutions

Let us first revisit KAT-based algebraic reasoning and highlight the challenges in extending the framework to the quantum setting. We assume a few self-explanatory quantum notations with detailed quantum preliminaries in Section 3.1.

KAT-based Reasoning. A typical reasoning framework based on KAT, similarly for NetKAT and GKAT, will establish that KAT models the targeted computation by showing

$$\vdash_{\mathrm{KAT}} e = f \quad \Leftrightarrow \quad \forall \mathrm{int}, \mathcal{K}_{\mathrm{int}}(e) = \mathcal{K}_{\mathrm{int}}(f), \qquad (1.2.1)$$

where $\mathcal{K}_{\mathrm{int}}$ is an interpretation mapping from expressions to a language (or semantic) model of the desired computation. In reasoning about while programs, one encodes them as KAT expressions as in Propositional Dynamic Logic [19]:

$$p; q := pq \qquad (1.2.2)$$

$$\textbf{if } b \textbf{ then } p \textbf{ else } q := bp + \overline{b}q \qquad (1.2.3)$$

$$\textbf{while } b \textbf{ do } p \textbf{ done} := (bp)^*\overline{b}, \qquad (1.2.4)$$

where b is a classical guard/test and $\overline{b}$ is its Boolean negation.

Intuitively, if one can derive the equivalence of encodings of two classical programs in KAT, then through the soundness direction ($\Rightarrow$), one can also establish the equivalence between the semantics of the original programs by applying an appropriate interpretation.

Quantum Branching. One *critical* difference between quantum and classical programs lies in the *branching* statement. The quantum branching statement,

$$\textbf{case } M[q] \rightarrow^{i} P_i \textbf{ end}, \qquad (1.2.5)$$

refers to a *probabilistic* procedure to execute branch P_i depending on the outcome of quantum measurement M on quantum variable q (of which the state is denoted by a density operator ρ). Consider the two-branching case (i=0,1), and let $M = \{M_0, M_1\}$ be the quantum measurement operators. Measurement M will *collapse* ρ to the state $\rho_0 = M_0\rho M_0^\dagger/\mathrm{tr}(M_0\rho M_0^\dagger)$ with probability $p_0 = \mathrm{tr}(M_0\rho M_0^\dagger)$, and the state $\rho_1 = M_1\rho M_1^\dagger/\mathrm{tr}(M_1\rho M_1^\dagger)$ with probability $p_1 = \mathrm{tr}(M_1\rho M_1^\dagger)$ respectively (here $\mathrm{tr}(\cdot)$ is the matrix trace). After the measurement M, the program will execute P_i on state ρ_i with probability p_i $(i = 0, 1)$.

There are two important differences between quantum and classical branching. The *first* is that quantum branching allows probabilistic choices over different branches. Even though random choices also appear in probabilistic programs, the probabilistic choices in quantum branching are due to quantum mechanics (i.e., measurements). In particular, their distributions are determined by the underlying quantum states and the corresponding quantum measurements, and hence *implicit* in the syntax of quantum programs, whereas specific probabilities are usually *explicitly* encoded in the syntax of probabilistic programs. Moreover, different quantum measurements do not necessarily commute with each other, which could hence lead to more complex probability distributions in quantum branching than ones allowed in classical probability theory and hence probabilistic programs.

The *second* difference lies in the different roles played by classical guards and quantum measurements in branching. Note that classical guards serve two functionalities simultaneously: (1) first, their values are used to choose the branches before the control; (2) second, they can also be deemed as property tests (i.e. logical propositions) on the state of the program after the control but before executing each branch. These two points might be so natural that one tends to forget that they are based on *an assumption that observing the guard won't change the state of the program*, which is also naturally held classically. The classical guards, when deemed as tests in KAT, enjoy further the Boolean algebraic properties so that they can be conveniently manipulated.

This natural assumption, however, fails to hold in quantum branching since quantum measurements will change underlying states in the branching statement. This is mathematically evident as we see ρ is collapsed to either ρ_0 or ρ_1 for different branches. Therefore, it is conceivable that quantum branching (and hence quantum programs) should refer to a different semantic model and quantum measurements should be deemed different from the tests in KAT.

Issues with directly adopting KAT/KA. Aforementioned differences make it hard to directly work with KAT/KA for quantum programs. First, there is a well-known issue when combining non-determinism, which is native to KAT, with probabilistic choices [43, 60], the latter of which is however essential in quantum branching. A similar issue also showed up in the probabilistic extension of NetKAT [20], which does not satisfy all the KAT rules, especially the *idempotent* law. One might wonder about the possibility of using GKAT [55], which is designed to mitigate this issue by restricting KAT with guarded structures. Unfortunately, the classical guarded structure modeled in GKAT is semantically different from quantum branching, which makes it hard to connect GKAT with appropriate quantum models.

Solution with NKA and NKAT. Our strategy is to work with the variant of KA without the idempotent law, namely, the *non-idempotent Kleene algebra* (NKA). This change will help model the probabilistic nature of quantum programs in a natural way, however, at the cost of losing properties implied by the idempotent law. Fortunately, thanks to the existing research on NKA [17, 40], many properties of KA are recovered in NKA for its applications to quantum programs.

Since there is no single "test" in quantum programs that can serve two purposes like classical guards, we simply separate the treatments for them. The branching functionality of quantum measurements can hence be expressed in NKA by treating them as normal program statements. Precisely, any quantum two-branching can be encoded as

$$m_0 p_0 + m_1 p_1, \tag{1.2.6}$$

where $m_{0/1}$ are encodings of measurements and $p_{0/1}$ are encodings of programs in each branch. Comparing with the classical encoding (1.2.3), $m_{0/1}$ no longer enjoy the Boolean algebraic properties and should be treated separately.

It turns out that many classical applications of KAT such as compiler optimization [36] and the proof of the normal form of **while**-programs [33] can be implemented in NKA for quantum programs with branching functionality only.

However, one needs to extend NKA to recover other applications of KAT which makes essential use of the proposition functionality of tests. A prominent example in KAT is its application to propositional Hoare logic [34]. Indeed, a typical Hoare triple $\{b\}p\{c\}$ asserts that whenever b holds before the execution of the program p, then if and when p halts, c will hold of the output state, where b, c are both tests in KAT leveraging their proposition functionality.

A similar triple $\{A\}P\{B\}$ is also used in quantum Hoare logic [63], where P is the quantum program and A, B become *quantum predicates* [15]. To encode quantum Hoare logic, we extend NKA with the "test", denoted NKAT, which mimics the behavior of quantum predicates following the effect algebra [22]. With quantum predicates, we develop a more delicate description of measurements in quantum branching, called *partitions*, which allow us to reason about the relationship among quantum branches caused by the same quantum measurement, e.g., the m_0 and m_1 branches in (1.2.6).

Quantum Path Model. One of our main technical contributions is the identification of the so-called *quantum path model*, a complete and sound semantic model for NKA. Namely,

$$\vdash_{\text{NKA}} e = f \quad \Leftrightarrow \quad \forall \text{int}, \mathcal{Q}_{\text{int}}(e) = \mathcal{Q}_{\text{int}}(f), \tag{1.2.7}$$

where $\mathcal{Q}_{\text{int}}$ is an interpretation mapping from NKA expressions to quantum path actions, which can be deemed as quantum evolution in the path integral formulation of quantum mechanics. $\mathcal{Q}_{\text{int}}$ will connect the NKA encoding of any quantum program P with its denotational semantics $[\![P]\!]$. [1]

The key motivation of the quantum path model is to address the infinity issue in NKA. For an intuitive understanding, one can deem any KA or NKA expression as a collection of potentially infinitely many traces, where "infinitely many" is caused by $*$ operations. In the case of KA, by the idempotent law, every single trace is either in or out of the collection. However, in the case of NKA, each trace is associated with a weight, which by itself could be infinite. To distinguish between nonequivalent NKA expressions, one needs to build a semantic model that can characterize a collection of weighted traces with potentially infinite weights. We also require the quantum nature of this semantic model for connection with the denotational semantics of quantum programs.

The path integral formulation becomes very natural in this regard: it formulates quantum evolution as the accumulative effect of a collection of evolutions on individual trajectories. Our quantum path model basically characterizes the accumulative quantum evolution over a collection of potentially infinite evolutions over individual traces. By identifying quantum path actions representing quantum predicates and quantum measurements in the quantum path model, a soundness theorem is proved for NKAT as well.

Quantum-Classical differences as exhibited in NKA and NKAT. The quantum-classical difference is not explicit in the syntax of NKA, as there is no special symbol for quantum measurements. This is also reflected in the proof of the completeness of NKA where an interpretation of essentially classical probabilistic processes is constructed (Remark 4.1).

However, the difference becomes explicit in NKAT: the two functionalities of the quantum guards are characterized separately by *effects* and *partitions*, in contrast with the classical guards in KAT. The general noncommutativity of quantum measurements in NKAT demonstrates its quantumness and distinguishes itself from any classical model.

1.3 Contributions

To our best knowledge, we contribute the first investigation of Kleene-like algebraic reasoning of quantum programs and demonstrate its feasibility. We introduce the non-idempotent Kleene algebra (NKA) and existing results on the semantic model of NKA in Section 2. Our contributions include:

- We illustrate the quantum path model and its relation with normal quantum superoperators in Section 3.
- We prove that the NKA axioms are sound and complete with respect to the quantum path model, given encodings of quantum programs in NKA and an appropriate interpretation of NKA to the quantum path model in Section 4.
- We demonstrate several applications of NKA for quantum programs, including: (1) the verification of optimization in quantum compilers (Section 5); (2) an algebraic equational proof of the quantum counterpart of the classic Böhm-Jacopini theorem [8] (Section 6).
- We extend NKA with the effect algebra to obtain the Non-idempotent Kleene Algebra with Tests (NKAT), which is proven sound for the quantum path model. We also encode the entire propositional quantum Hoare logic as NKAT theorems in light of Kozen [34] (Section 7).

[1]Since we relate NKA to quantum models which imply the probabilistic feature inherently, there is no need to explicitly add probability to NKA.

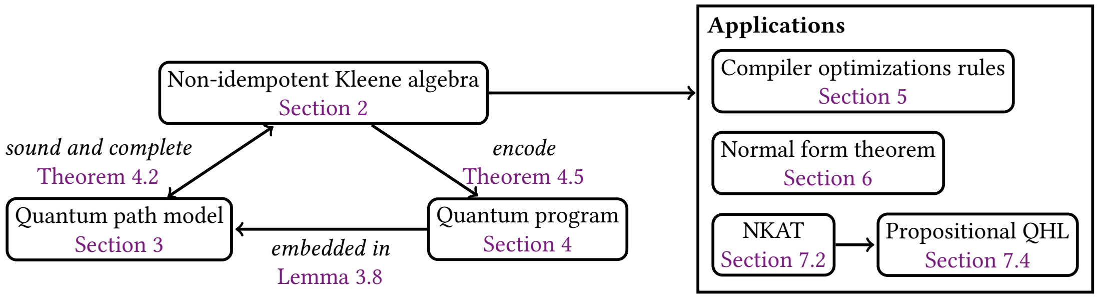

Figure 1. The structure and main results of this paper.

Main Theorem. Our main theorem presented below formally guarantees that quantum program equivalences are implied if we can algebraically derive the corresponding NKA theorems. This approach is similar to deriving classical program equivalence via KAT.

Theorem 1.1. *Given two quantum programs P, Q and subprogram pairs $\{\langle P_i, Q_i\rangle\}$ where $[\![P_i]\!] = [\![Q_i]\!]$, if Horn theorem*

$$\vdash_{\text{NKA}} \left(\bigwedge_i \text{Enc}(P_i) = \text{Enc}(Q_i)\right) \rightarrow \text{Enc}(P) = \text{Enc}(Q)$$

is derivable, then we have $[\![P]\!] = [\![Q]\!]$. Here Enc *is the encoding of quantum program in a similar manner of* (1.2.2)-(1.2.4).

We display the essential concepts leading to this theorem in Figure 1, illustrating how our efforts in later sections connect to it, and its applications and extensions.

Related Works. It is worthwhile comparing quantum algebraic reasoning based on NKA with other techniques on quantum program analysis, e.g., quantum Hoare logic [63]. As we see, classical algebraic reasoning is extremely good at certain tasks (e.g, equational proofs). However, since it abstracts away a lot of semantic information, it cannot tell about detailed specifications on the state of programs, which can otherwise be reasoned by Hoare logic [28].

Our quantum algebraic reasoning inherits the advantages and disadvantages of its classical counterpart. It allows elegant applications in Section 5 & 6, which is very hard (e.g., involving exponential-size matrices) to solve with the quantum Hoare logic [63] or its relational variants [6, 59]. However, it cannot replace quantum Hoare logic to reason about, e.g., specifications on the state of quantum programs either.

A recent result of quantum abstract interpretation [68] contributes to another promising approach to verifying quantum assertions with succinct proofs, although its applicability and technique are incomparable to ours.

There are many other verification tools developed for quantum programs. Hietala et al. [27] built VOQC, an infrastructure for quantum circuits in Coq with numerous verified programs and compiler optimization rules. Another theory for equational reasoning of quantum circuits is introduced in [56]. They serve as good complements of our framework when loops are absent.

Future Directions. One interesting question is the automation related to NKA, e.g., through co-algebra and bi-simulation techniques, in light of [9, 35, 54, 55]. This could lead to efficient symbolic algorithms for algebraic reasoning of quantum programs in light of [48]. Kiefer et al. [30] proposed an algorithm checking $\mathbb{Q}-$weighted automata equivalences, which works for NKA when no infinity presents.

Another direction is to include quantum-specific rules to NKA to ease the expression of practical quantum applications. For example, one may embed unitary superoperators into NKA as a group to encode their reversibility.

Given the promising applications of KAT in network programming (e.g., NetKAT [20]), an exciting opportunity is to investigate the possibility of a quantum version of NetKAT in the software-defined model of the emerging quantum internet (e.g., [11, 38]) based on our work.

The appendix appears in our extended version [47] on arXiv.

2 Non-idempotent Kleene Algebra

In this section, we introduce the theory of a Kleene algebraic system without the idempotent law, which is called non-idempotent Kleene algebra (NKA).

We inherit Kozen's axiomatization for Kleene algebra (KA) in [32] with several weakenings.

Definition 2.1. *A non-idempotent Kleene algebra (NKA) is a 7-tuple $(\mathcal{K}, +, \cdot, *, \leq, 0, 1)$, where $+$ and $\cdot$ are binary operations, $*$ is a unary operation, and $\leq$ is a binary relation. It satisfies the axioms in Figure 3.*

The most essential weakening is the deletion of the idempotent law. The partial order in KA cannot directly fit in the scenario when the idempotent law is absent. We hence generalize the KA partial order to any partial order that is preserved by $+$ and $\cdot$. Therefore, $*$ also preserves this partial order. Moreover, we did not include the symmetric fixed point inequality $1 + p^*p \leq p^*$ because it is derivable by other axioms, both in KA and in NKA [17].

Definition 2.2. *For an alphabet Σ, an expression over Σ is inductively defined by:*

$$e ::= 0 \mid 1 \mid a \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid e_1^*, \qquad (2.0.1)$$

$1 + pp^* = 1 + p^*p = p^*$	(fixed-point)	$(pq)^*p = p(qp)^*$	(sliding)
$p \le q \to p^* \le q^*$	(monotone-star)	$(p+q)^* = (p^*q)^*p^* = p^*(qp^*)^*$	(denesting)
$1 + p(qp)^*q = (pq)^*$	(product-star)	$0 \le p$	(positivity)

(a) Commonly used theorems of NKA

$(pp)^*(1+p) = p^*$	(unrolling)
$pq = qp \to p^*q = qp^*$	(swap-star)
$pq = rp \to pq^* = r^*p$	(star-rewrite)

(b) Several theorems of NKA for applications

Figure 2. Derivable formulae in NKA.

where $a \in \Sigma$. We denote all the expressions over Σ by Exp_Σ.

A Horn formula ϕ is defined as the form $(\bigwedge_i e_i \le f_i) \to e \le f$. One may also substitute equation for inequality in ϕ since $e = f \leftrightarrow e \le f \wedge f \le e$.

We write $\vdash_{\mathrm{NKA}} \phi$ if ϕ is derivable in NKA with equational logic. Any derivable formula in NKA is a theorem *of NKA.*

Apparently, every theorem in NKA is derivable in KA, since the partial order in KA is monotone. The reverse direction is not true in general. Indeed, the idempotent law, for example, is nowhere derivable from the NKA axioms. It is thus natural to ask what important theorems in KA are still derivable in NKA. We provide affirmative answers to many of them in the following. (Proofs in the appendix [47].)

Axioms of KA	**Axioms of NKA**
Semiring Laws	Semiring Laws:
$p + (q + r) = (p + q) + r$;	$p + (q + r) = (p + q) + r$;
$p + q = q + p$;	$p + q = q + p$;
$p + 0 = p$;	$p + 0 = p$;
$p(qr) = (pq)r$;	$p(qr) = (pq)r$;
$1p = p1 = p$;	$1p = p1 = p$;
$0p = p0 = 0$;	$0p = p0 = 0$;
$p(q + r) = pq + pr$;	$p(q + r) = pq + pr$;
$(p + q)r = pr + qr$;	$(p + q)r = pr + qr$;
$p + p = p$;	
Partial Order Laws	Partial Order Laws
$p \le q \leftrightarrow p + q = q$;	$p \le p$;
	$p \le q \wedge q \le p \to p = q$;
	$p \le q \wedge q \le r \to p \le r$;
	$p \le q \wedge r \le s \to p + r \le q + s$;
	$p \le q \wedge r \le s \to pr \le qs$;
Star Laws	Star Laws
$1 + pp^* \le p^*$;	$1 + pp^* \le p^*$;
$q + pr \le r \to p^*q \le r$;	$q + pr \le r \to p^*q \le r$;
$q + rp \le r \to qp^* \le r$;	$q + rp \le r \to qp^* \le r$;

Figure 3. Axioms of KA and NKA. Axioms marked in blue (red) only present in NKA (KA).

Lemma 2.3. *The following formulae are derivable in NKA.*

1. *The formulae in Figure 2a due to [17].*
2. *The formulae in Figure 2b.*

It is known that NKA also has a natural semantic model, called rational power series, which is a special class of formal power series over $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$. We present a brief introduction to them in the appendix [47] for interested readers.

Remark 2.1 (Complexity related to NKA)**.** *Bloom and Ésik [7] have proposed an algorithm to determine the equivalence of two rational power series, so the equational theory of NKA is decidable. Meanwhile, a subset $1^*\mathcal{K} = \{1^*p : p \in \mathcal{K}\}$ satisfies the Kleene algebra axioms, and the equational theory of KA is PSpace-complete [57], thus equational theory of NKA is also PSpace-hard. However, by linking formal power series to weighted finite automata, Eilenberg [16] shows that it is undecidable whether a given inequality $e \le f$ holds in NKA.*

3 Quantum Path Model

To address the infinity issue, we introduce a generalization of quantum superoperators in this section, named quantum path model, a sound model of NKA. We include detailed quantum preliminaries in Section 3.1, introduce extended positive operators as a generalization of quantum states in Section 3.2, define the quantum path model as an analog of the path integral in quantum mechanics in Section 3.3, and embed quantum superoperators in the quantum path model in Section 3.4. We recommend that first-time readers skip technical construction details in this section.

3.1 Quantum Preliminaries

We review basic notations from quantum information that are used in this paper. Curious readers should refer to [44, 61] for more details.

An n-dimensional Hilbert space $\mathcal{H}$ is essentially the space $\mathbb{C}^n$ of complex vectors. We use Dirac's notation, $|\psi\rangle$, to denote a complex vector in $\mathbb{C}^n$. The inner product of $|\psi\rangle$ and $|\varphi\rangle$ is denoted by $\langle\psi|\varphi\rangle$, which is the product of the Hermitian conjugate of $|\psi\rangle$, denoted by $\langle\psi|$, and the vector $|\varphi\rangle$.

Linear operators between n-dimensional Hilbert spaces are represented by $n \times n$ matrices. For example, the zero operator $O_\mathcal{H}$ and the identity operator $I_\mathcal{H}$ can be identified by the zero matrix and the identity matrix on $\mathcal{H}$. The Hermitian conjugate of operator A is denoted by $A^\dagger$. Operator A is *positive semidefinite* if for all vectors $|\psi\rangle \in \mathcal{H}$, $\langle\psi|A|\psi\rangle \ge 0$.

The set of positive semidefinite operators over $\mathcal{H}$ is denoted by $\mathcal{PO}(\mathcal{H})$. This gives rise to the *Löwner order* $\sqsubseteq$ among operators: $A \sqsubseteq B \Leftrightarrow B - A$ is positive semidefinite.

A *density operator* ρ is a positive semidefinite operator $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ where $\sum_i p_i = 1, p_i > 0$. A special case $\rho = |\psi\rangle\langle\psi|$ is conventionally denoted as $|\psi\rangle$. A positive semidefinite operator ρ on $\mathcal{H}$ is a *partial* density operator if $\text{tr}(\rho) \leq 1$, where $\text{tr}(\rho)$ is the matrix trace of ρ. The set of partial density operators is denoted by $\mathcal{D}(\mathcal{H})$.

The evolution of a quantum system can be characterized by a *completely-positive* and *trace-non-increasing* linear superoperator $\mathcal{E}$[2], which is a mapping from $\mathcal{D}(\mathcal{H})$ to $\mathcal{D}(\mathcal{H}')$ for Hilbert spaces $\mathcal{H}, \mathcal{H}'$. We denote the set of such superoperators by $\mathcal{QC}(\mathcal{H}, \mathcal{H}')$. The special case when $\mathcal{H}' = \mathcal{H}$ is denoted by $\mathcal{QC}(\mathcal{H})$.

For two superoperators $\mathcal{E}_1, \mathcal{E}_2 \in \mathcal{QC}(\mathcal{H})$, the composition is defined as $(\mathcal{E}_1 \circ \mathcal{E}_2)(\rho) = \mathcal{E}_2(\mathcal{E}_1(\rho))$. If there exists $\mathcal{E}$ and $\mathcal{E}_i \in \mathcal{QC}(\mathcal{H})$ satisfying $\mathcal{E}(\rho) = \sum_i \mathcal{E}_i(\rho)$ for every $\rho \in \mathcal{PO}(\mathcal{H})$, then we define $\mathcal{E}$ as $\sum_i \mathcal{E}_i$. For every superoperator $\mathcal{E} \in \mathcal{QC}(\mathcal{H}, \mathcal{H}')$, by [39] there exists a set of Kraus operators $\{E_k\}_k$ such that $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$ for any input $\rho \in \mathcal{D}(\mathcal{H})$. The Schrödinger-Heisenberg *dual* of a superoperator $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$ is $\mathcal{E}^\dagger(\rho) = \sum_k E_k^\dagger \rho E_k$.

A quantum *measurement* on a system over Hilbert space $\mathcal{H}$ can be described by a set of linear operators $\{M_m\}_m$ where $\sum_m M_m^\dagger M_m = I_{\mathcal{H}}$. The measurement outcome m is observed with probability $p_m = \text{tr}(M_m \rho M_m^\dagger)$ for each m, which will collapse the pre-measure state ρ to $\mathcal{M}_m(\rho) = M_m \rho M_m^\dagger / p_m$. A quantum measurement is *projective* if $M_i M_j = M_i$ if $i = j$ and $O_{\mathcal{H}}$ otherwise. Namely, all M_i are projective operators orthogonal to each other.

3.2 Extended Positive Operators

The set $\mathcal{PO}(\mathcal{H})$ does not contain any infinity. We need to incorporate different infinities into it to distinguish different path sets which may lead to different divergent summations.

Definition 3.1. *A series of $\mathcal{PO}(\mathcal{H})$ is a countable multiset of $\mathcal{PO}(\mathcal{H})$, and can be written as $\biguplus_{i\in I} \rho_i$, where I is a countable index set. Symbol $\biguplus_{i\in I}$ enumerates every element ρ_i in the multiset. The set of series of $\mathcal{PO}(\mathcal{H})$ is denoted by $\mathcal{S}(\mathcal{H})$.*

The union *of countably many series is denoted by:*

$$\biguplus_{i\in I} \left(\biguplus_{j\in J_i} \rho_{ij} \right) = \biguplus_{(i,j): i\in I, j\in J_i} \rho_{ij}. \tag{3.2.1}$$

Note $\biguplus_{i\in I} \biguplus_{j\in J_i} \rho_{ij} \in \mathcal{S}(\mathcal{H})$ since the index set is countable.

A binary relation $\lesssim$ *over $\mathcal{S}(\mathcal{H})$ is defined by: $\biguplus_{i\in I} \rho_i \lesssim \biguplus_{j\in J} \sigma_j$ if and only if for every $\epsilon > 0$ and finite $I' \subseteq I$, there exists a finite $J' \subseteq J$, such that*

$$\sum_{i\in I'} \rho_i \sqsubseteq \epsilon I_{\mathcal{H}} + \sum_{j\in J'} \sigma_j. \tag{3.2.2}$$

[2]A superoperator $\mathcal{E}$ is *positive* if it maps from $\mathcal{D}(\mathcal{H})$ to $\mathcal{D}(\mathcal{H}')$ for Hilbert spaces $\mathcal{H}, \mathcal{H}'$. It is *completely-positive* if for any Hilbert space $\mathcal{A}$, the superoperator $\mathcal{E} \otimes I_{\mathcal{A}}$ is positive. It is *trace-non-increasing* if for any initial state $\rho \in \mathcal{D}(\mathcal{H})$, the final state $\mathcal{E}(\rho) \in \mathcal{D}(\mathcal{H}')$ satisfies $\text{tr}(\mathcal{E}(\rho)) \leq \text{tr}(\rho)$.

We induce another binary relation $\sim$ from $\lesssim$ on $\mathcal{S}(\mathcal{H})$ by:

$$\biguplus_{i\in I} \rho_i \sim \biguplus_{j\in J} \sigma_j \quad \Leftrightarrow \quad \biguplus_{i\in I} \rho_i \lesssim \biguplus_{j\in J} \sigma_j \ \wedge \ \biguplus_{j\in J} \sigma_j \lesssim \biguplus_{i\in I} \rho_i.$$

Symbol $\biguplus_{i\in I}$ is employed to distinguish the series from the normal summation $\sum_{i\in I}$ over $\mathcal{PO}(\mathcal{H})$. We will build connections between these two notions so that $\biguplus_{i\in I}$ can readily help us in the analysis of convergence, and more.

We represent a finite series by enumerating its elements. Like a series with one element $O_{\mathcal{H}}$, we denote it by $\{\!|O_{\mathcal{H}}|\!\}$.

The definition of $\lesssim$ aims at a generalization to the Löwner order in $\mathcal{S}(\mathcal{H})$ that distinguishes the different infinities while preserving relations like $\{\!|I_{\mathcal{H}}|\!\} \lesssim \biguplus_{i>0} \frac{1}{2^i} I_{\mathcal{H}}$, whose correspondence in $\mathcal{PO}(\mathcal{H})$ holds.

Lemma 3.2. *$\lesssim$ is a preorder, so $\sim$ is an equivalence relation.*

The proof of this lemma along with several basic facts about $\mathcal{S}(\mathcal{H})$ is in the appendix [47].

Definition 3.3. *We define the extended positive operators $\mathcal{PO}_\infty(\mathcal{H}) = \mathcal{S}(\mathcal{H})/\!\sim$ as the set of equivalence classes of $\sim$. Let the equivalence class including $\biguplus_{i\in I} \rho_i$ be*

$$\left[\biguplus_{i\in I} \rho_i \right] = \left\{ \biguplus_{j\in J} \sigma_j \ \middle|\ \biguplus_{j\in J} \sigma_j \sim \biguplus_{i\in I} \rho_i \right\}, \tag{3.2.3}$$

where on the right hand side is a set of series.

A partial order $\leq$ over $\mathcal{PO}_\infty(\mathcal{H})$ is induced from the preorder $\lesssim$ over $\mathcal{S}(\mathcal{H})$ by:

$$\left[\biguplus_{i\in I} \rho_i \right] \leq \left[\biguplus_{j\in J} \sigma_j \right] \Leftrightarrow \biguplus_{i\in I} \rho_i \lesssim \biguplus_{j\in J} \sigma_j. \tag{3.2.4}$$

We define countable summation over $\mathcal{PO}_\infty(\mathcal{H})$ from the union in $\mathcal{S}(\mathcal{H})$ by

$$\sum_{i\in I} \left[\biguplus_{j\in J_i} \rho_{ij} \right] = \left[\biguplus_{i\in I} \biguplus_{j\in J_i} \rho_{ij} \right]. \tag{3.2.5}$$

The summation defined above is independent of the choices of $\biguplus_{j\in J_i} \rho_{ij}$, whose reasoning is in the appendix [47].

We slightly abuse notation, writing $[\rho]$ to represent $[\{\!|\rho|\!\}]$ for $\rho \in \mathcal{PO}(\mathcal{H})$. A frequently used case of (3.2.5) is to write the equivalence class of a series as

$$\left[\biguplus_{i\in I} \rho_i \right] = \sum_{i\in I} [\rho_i], \tag{3.2.6}$$

where we can intuitively deem the countable summation over $\mathcal{PO}_\infty(\mathcal{H})$ as a generalized summation over $\mathcal{PO}(\mathcal{H})$. For example, we have $\sum_{i>0} \left[\frac{1}{2^i} I_{\mathcal{H}} \right] = \left[\sum_{i>0} \frac{1}{2^i} I_{\mathcal{H}} \right] = [I_{\mathcal{H}}]$. The reasoning is in the appendix [47].

Remark 3.1. *$\mathcal{PO}(\mathcal{H})$ is embedded in $\mathcal{PO}_\infty(\mathcal{H})$ by $\rho \mapsto [\rho]$ as finite positive operators. Besides these, $\mathcal{PO}_\infty(\mathcal{H})$ contains distinguishable divergent summations unattainable by $\mathcal{PO}(\mathcal{H})$: e.g., $\sum_{i>0} [|0\rangle\langle 0|]$ is different from $\sum_{i>0} [|1\rangle\langle 1|]$, and less than $\sum_{i>0} [I_{\mathcal{H}_2}]$. These divergent summations are leveraged to depict the domain and the range of our extended quantum superoperators.*

3.3 Quantum Actions

We are now ready to introduce quantum actions, a generalization of superoperators in the quantum path model, inspired by the path integral formulation of quantum mechanics.

Definition 3.4. *A* quantum action*, or* action *for simplicity, over $\mathcal{PO}_\infty(\mathcal{H})$ is a mapping from $\mathcal{PO}_\infty(\mathcal{H})$ to $\mathcal{PO}_\infty(\mathcal{H})$.*

A quantum action $\mathcal{A}$ is linear *if for series $\sum_{j\in J_i}\left[\rho_{ij}\right]$,*

$$\mathcal{A}\left(\sum_{i\in I}\sum_{j\in J_i}\left[\rho_{ij}\right]\right)=\sum_{i\in I}\mathcal{A}\left(\sum_{j\in J_i}\left[\rho_{ij}\right]\right). \quad (3.3.1)$$

A quantum action $\mathcal{A}$ is monotone *if for any two series $\sum_{i\in I}[\rho_i]\le\sum_{j\in J}\left[\sigma_j\right]$,*

$$\mathcal{A}\left(\sum_{i\in I}[\rho_i]\right)\le\mathcal{A}\left(\sum_{j\in J}\left[\sigma_j\right]\right). \quad (3.3.2)$$

We denote the set of linear and monotone quantum actions over $\mathcal{PO}_\infty(\mathcal{H})$ by $\mathcal{P}(\mathcal{H})$ as the set of quantum path actions.

The zero action $O_\mathcal{H}$ maps everything to $[O_\mathcal{H}]$, and the identity action is denoted by $\mathcal{I}_\mathcal{H}$.

A physical interpretation of quantum path actions in $\mathcal{P}(\mathcal{H})$ is the collection of quantum evolution along a single or many possible trajectories of the underlying system. Thus, one can readily define the composition and the sum of quantum path actions, as the concatenation and the union of trajectories.

Definition 3.5. *We define the operations in $\mathcal{P}(\mathcal{H})$ by:*

$$\left(\sum_{i\in I}\mathcal{A}_i\right)\left(\sum_{j\in J}\left[\rho_j\right]\right)=\sum_{i\in I}\mathcal{A}_i\left(\sum_{j\in J}\left[\rho_j\right]\right), \quad (3.3.3)$$

$$(\mathcal{A}_1;\mathcal{A}_2)\left(\sum_{j\in J}\left[\rho_j\right]\right)=\mathcal{A}_2\left(\mathcal{A}_1\left(\sum_{j\in J}\left[\rho_j\right]\right)\right), \quad (3.3.4)$$

$$\mathcal{A}^*=\sum_{i\ge 0}\mathcal{A}^i. \quad (3.3.5)$$

Here $\mathcal{A}^i=\mathcal{I}_\mathcal{H};\mathcal{A};\mathcal{A};\cdots;\mathcal{A}$ where $\mathcal{A}$ repeats i times.

Additionally, we define $\mathcal{A}_1\diamond\mathcal{A}_2=\mathcal{A}_2;\mathcal{A}_1$.

A point-wise partial order $\le$ in $\mathcal{P}(\mathcal{H})$ is induced point-wisely: $\mathcal{A}_1\le\mathcal{A}_2$ if and only if

$$\forall\sum_{i\in I}[\rho_i],\ \mathcal{A}_1\left(\sum_{i\in I}[\rho_i]\right)\le\mathcal{A}_2\left(\sum_{i\in I}[\rho_i]\right). \quad (3.3.6)$$

Our main result is that $\mathcal{P}(\mathcal{H})$ with the above partial order and operations satisfies the axioms of NKA. The proof is postponed to the appendix [47]. Since infinite summations are well-defined over quantum path actions, any NKA derivation safely induces a derivation over quantum path actions without worrying about the infinity issue.

Theorem 3.6. *The NKA axioms are sound for the* quantum path model*, defined by $(\mathcal{P}(\mathcal{H}),+,\ ;,*,\le,O_\mathcal{H},\mathcal{I}_\mathcal{H})$. Here $+$ is the $\sum_i$ operation restricted on two operands.*

3.4 Embedding of $\mathcal{QC}(\mathcal{H})$ in $\mathcal{P}(\mathcal{H})$

We mentioned the intuition that quantum path actions are generalizations of quantum superoperators in the quantum path model. We now make it precise by building an embedding from quantum superoperators to quantum path actions (and hence the quantum path model), which allows us to prove superoperator equations via NKA theorems.

Definition 3.7. Path lifting *is a mapping from $\mathcal{E}\in\mathcal{QC}(\mathcal{H})$ to a quantum path action $\langle\mathcal{E}\rangle^\uparrow:\sum_{i\in I}[\rho_i]\mapsto\sum_{i\in I}[\mathcal{E}(\rho_i)]$.*

$\langle\mathcal{E}\rangle^\uparrow$ is well-defined (it does not depend on the choices of $\sum_{i\in I}[\rho_i]$). The reasoning is in the appendix [47].

The path lifting embeds $\mathcal{QC}(\mathcal{H})$ in $\mathcal{P}(\mathcal{H})$ by the following lemma, whose proof is routine and in the appendix [47].

Lemma 3.8. *The path lifting has the following properties:*

(i) $\langle\mathcal{E}\rangle^\uparrow\in\mathcal{P}(\mathcal{H})$, for $\mathcal{E}\in\mathcal{QC}(\mathcal{H})$.
(ii) $\mathcal{E}_1=\mathcal{E}_2\Leftrightarrow\langle\mathcal{E}_1\rangle^\uparrow=\langle\mathcal{E}_2\rangle^\uparrow$, for $\mathcal{E}_1,\mathcal{E}_2\in\mathcal{QC}(\mathcal{H})$.
(iii) operations $\circ$ and $\sum_i$ (when defined) in $\mathcal{QC}(\mathcal{H})$ are preserved by path lifting as $;$ and $\sum_i$ operations in $\mathcal{P}(\mathcal{H})$.

4 Quantum Interpretation and Quantum Programs

In this section, we link expressions, quantum path actions and quantum programs by quantum interpretation (Section 4.1) and encoding (Section 4.2).

4.1 Quantum Interpretation

We endow equations in NKA with quantum interpretations.

Definition 4.1. *A* quantum interpretation setting *over an alphabet Σ is a pair* $\mathrm{int}=(\mathcal{H},\mathrm{eval})$ *where*

1. *$\mathcal{H}$ is a finite dimensional Hilbert space.*
2. $\mathrm{eval}:\Sigma\to\mathcal{QC}(\mathcal{H})$ *is a function to interpret symbols.*

The quantum interpretation $\mathcal{Q}_{\mathrm{int}}$ *w.r.t. a quantum interpretation setting* int *is a mapping from* Exp_Σ *to $\mathcal{P}(\mathcal{H})$ where*

$$\mathcal{Q}_{\mathrm{int}}(0)=O_\mathcal{H},\qquad \mathcal{Q}_{\mathrm{int}}(e+f)=\mathcal{Q}_{\mathrm{int}}(e)+\mathcal{Q}_{\mathrm{int}}(f),$$
$$\mathcal{Q}_{\mathrm{int}}(1)=\mathcal{I}_\mathcal{H},\qquad \mathcal{Q}_{\mathrm{int}}(e\cdot f)=\mathcal{Q}_{\mathrm{int}}(e);\mathcal{Q}_{\mathrm{int}}(f),$$
$$\mathcal{Q}_{\mathrm{int}}(a)=\langle\mathrm{eval}(a)\rangle^\uparrow,\qquad \mathcal{Q}_{\mathrm{int}}(e^*)=\mathcal{Q}_{\mathrm{int}}(e)^*.$$

Here $a\in\Sigma$, and $\langle\mathrm{eval}(a)\rangle^\uparrow$ is the path lifting of $\mathrm{eval}(a)$.

Theorem 4.2. *The axioms of NKA are sound and complete w.r.t. the quantum interpretation. That is, for any $e,f\in\mathrm{Exp}_\Sigma$,*

$$\vdash_{\mathrm{NKA}} e=f \quad\Leftrightarrow\quad \forall\mathrm{int},\mathcal{Q}_{\mathrm{int}}(e)=\mathcal{Q}_{\mathrm{int}}(f). \quad (4.1.1)$$

The soundness comes directly from Theorem 3.6. The completeness proof makes use of formal power series and is postponed to the appendix [47]. This result indicates that equations of NKA are all possible tautologies when atomic symbols are interpreted as any (lifted) quantum superoperator. These equations and interpretations do not necessarily correspond to quantum programs, so further exploitation of algebraic structures specifically for quantum programs is possible.

Remark 4.1. *The completeness proof constructs interpretations with probabilistic processes only. It suggests that quantum processes have similar algebraic behaviors to probabilistic processes when probabilities are implicit (abstracted inside atomic* *operations). This is valid when measurements are not distinguished from other processes. We will discuss additional axioms for quantum measurements in Section 7.*

Most of the derived rules in our applications rely on external hypotheses aside from the NKA axioms. A formula with inequalities as hypotheses is called a Horn clause. We present the relation of the Horn theorems of NKA and quantum interpretations by the following theorem.

Corollary 4.3. *For expressions* $\{e_i\}_{i=1}^n, \{f_i\}_{i=1}^n \subset \mathrm{Exp}_\Sigma$ *and* $e, f \in \mathrm{Exp}_\Sigma$, *if*

$$\vdash_{\mathrm{NKA}} \left(\bigwedge_{i=1}^{n} e_i \leq f_i\right) \to e \leq f, \tag{4.1.2}$$

and $\mathrm{int} = (\mathcal{H}, \mathrm{eval})$ *satisfies* $Q_{\mathrm{int}}(e_i) \leq Q_{\mathrm{int}}(f_i)$ *for* $1 \leq i \leq n$, *then* $Q_{\mathrm{int}}(e) \leq Q_{\mathrm{int}}(f)$.

Note that the inequalities above can be replaced by equations, using the fact that $p = q \leftrightarrow p \leq q \wedge q \leq p$.

Proof. The proof comes from Theorem 3.6 similarly. Along the derivation of $e \leq f$, we apply the NKA axioms and premises $e_i \leq f_i$ for $1 \leq i \leq n$. The soundness of $e \leq f$ comes from the soundness of NKA axioms, proved in Theorem 3.6, and the soundness of each premises, provided by the assumption $Q_{\mathrm{int}}(e_i) \leq Q_{\mathrm{int}}(f_i)$ for each $e_i \leq f_i$. □

4.2 Encoding of Quantum Programs

The syntax of a *quantum* ***while*** *program*, also called a program for simplicity, P is defined as follows. [3]

$$P \ ::= \ \mathbf{skip} \ | \ \mathbf{abort} \ | \ q := |0\rangle \ | \ \overline{q} := U[\overline{q}] \ | \ P_1; P_2 \ | \\ \mathbf{case}\ M[\overline{q}] \xrightarrow{i} P_i \ \mathbf{end} \ | \ \mathbf{while}\ M[\overline{q}] = 1 \ \mathbf{do}\ P_1 \ \mathbf{done}.$$

The denotational semantics of P is a quantum superoperator, denoted by $[\![P]\!]$. Ying [64] proves that:

$$[\![\mathbf{skip}]\!](\rho) = \rho, \quad \left[\!\!\left[\mathbf{case}\ M[\overline{q}] \xrightarrow{i} P_i\ \mathbf{end}\right]\!\!\right] = \sum_i \mathcal{M}_i \circ [\![P_i]\!],$$

$$[\![\mathbf{abort}]\!](\rho) = O_{\mathcal{H}}, \quad [\![q := |0\rangle]\!](\rho) = \sum_i |0\rangle_q\langle i|\rho|i\rangle_q\langle 0|,$$

$$[\![P_1; P_2]\!] = [\![P_1]\!] \circ [\![P_2]\!], \quad [\![\overline{q} := U[\overline{q}]]\!](\rho) = U_{\overline{q}}\rho U_{\overline{q}}^\dagger,$$

$$[\![\mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ P\ \mathbf{done}]\!] = \sum_{n\geq 0}((\mathcal{M}_1 \circ [\![P]\!])^n \circ \mathcal{M}_0),$$

where for a quantum measurement $\{M_i\}_{i\in I}$, $\mathcal{M}_i$ is defined by $\mathcal{M}_i(\rho) = M_i\rho M_i^\dagger$. Both $\circ$ and $\sum_i$ are operations over quantum superoperators.

We formally define how to encode a quantum program as an expression, and how to recover the denotational semantics of a quantum program from an expression.

Definition 4.4. *An* encoder setting *is a mapping E from a finite subset of $QC(\mathcal{H})$ to Σ, that assigns a unique symbol in Σ to the elementary superoperators (qubit resetting, unitary application, and measurement branches) in the target programs.*

The encoder Enc *of a program to* Exp_Σ *with respect to an encoder setting E is defined inductively by:*

$$\mathrm{Enc}(\mathbf{skip}) = 1; \qquad \mathrm{Enc}(q := |0\rangle) = E([\![q := |0\rangle]\!]);$$

$$\mathrm{Enc}(\mathbf{abort}) = 0; \qquad \mathrm{Enc}(\overline{q} := U[\overline{q}]) = E([\![\overline{q} := U[\overline{q}]]\!]);$$

$$\mathrm{Enc}(P_1; P_2) = \mathrm{Enc}(P_1) \cdot \mathrm{Enc}(P_2);$$

$$\mathrm{Enc}(\mathbf{case}\ M[\overline{q}] \xrightarrow{i} P_i\ \mathbf{end}) = \sum_i E(\mathcal{M}_i) \cdot \mathrm{Enc}(P_i);$$

$$\mathrm{Enc}(\mathbf{while}\, M[\overline{q}] = 1\, \mathbf{do}\ P\ \mathbf{done}) = (E(\mathcal{M}_1)\cdot\mathrm{Enc}(P))^* \cdot E(\mathcal{M}_0),$$

where Σ_i *in (4.2.1) is an abbreviation of expression summation.*

Theorem 4.5. *For any quantum program P and encoder setting E, let* $\mathrm{int} = (\mathcal{H}, E^{-1})$, *where* E^{-1} *maps back the unique symbol for an elementary superoperator. Then*

$$Q_{\mathrm{int}}(\mathrm{Enc}(P)) = \langle[\![P]\!]\rangle^{\uparrow}. \tag{4.2.1}$$

A full proof by induction on P is in the appendix [47].

Note that in real applications, we usually define the encoder setting E jointly for multiple programs $\{P_i\}$ for technical convenience and easy comparison.

Now we have all the ingredients for Theorem 1.1.

Proof of Theorem 1.1. We have constructed the quantum path model and proved it a sound model of NKA in Theorem 3.6, leading to the soundness of Horn theorems by Corollary 4.3. We also show an embedding of quantum superoperators into quantum path actions in Lemma 3.8.(ii), so Horn theorems are interpreted as quantum superoperator equivalences. For each quantum program, we encode it with a symbolic expression whose interpretation corresponds to its denotational semantics, according to Theorem 4.5. Hence, if the NKA equivalence of quantum programs' encoding is derivable, the equivalence of their denotational semantics is induced. □

In the next sections, we show applications of Theorem 1.1.

5 Validation of Quantum Compiler Optimizing Rules

We demonstrate a few quantum compiler optimizing rules and their validation in NKA, in light of a similar application of KAT [36]. Note that many classical compiler optimizing rules do not hold or make sense in the quantum setting. We

[3]The **skip** statement does nothing and terminates. The **abort** statement announces that the program fails, and halts the program without any result. Statement $q := |0\rangle$ resets the register q to $|0\rangle$, and $\overline{q} := U[\overline{q}]$ applies a unitary operation on register set $\overline{q}$. These four statements' denotational semantics are called elementary superoperators. Note that there is no assignment statement due to the quantum no-cloning theorem [62]. The loop **while** $M[\overline{q}] = 1$ **do** P_1 **done** executes repeatedly. Each time it measures $\overline{q}$ by M. If the measurement result is 1, it executes P_1 and then starts over. Otherwise, it terminates. When there are only two branches, we define syntax sugar **if** $M[\overline{q}] = 1$ **then** P_1 **else** P_2 as an alternative to **case** $M[\overline{q}] \to^i P_i$ **end**. Moreover, if $P_2 \equiv$ **skip**, we write **if** $M[\overline{q}] = 1$ **then** P_1.

have carefully selected those rules with reasonable quantum counterparts, as well as quantum-specific rules found in real quantum applications.

The validation of quantum program equivalence via NKA consists of three steps: (1) *program encoding*: encode the programs as expressions over an alphabet; (2) *condition formulation*: identify necessary hypotheses and construct a formula that encodes hypotheses and target equation; (3) *NKA derivation*: derive the formula with the NKA axioms.

5.1 Loop Unrolling

Consider programs Unrolling1 and Unrolling2 in Figure 4 with a program P and a projective measurement M.

Program Encoding: We encode the two programs by expressions $\mathrm{Enc}(\textsc{Unrolling1}) = (m_0p)^*m_1$ and $\mathrm{Enc}(\textsc{Unrolling2}) = (m_0p(m_0p+m_1\cdot 1))^*m_1$. The encoder setting is inferred easily.

Condition Formulation: Because M is a projective measurement, $\mathcal{M}_1 \circ \mathcal{M}_1 = \mathcal{M}_1$ and $\mathcal{M}_1 \circ \mathcal{M}_0 = \mathcal{O}_{\mathcal{H}}$ can be encoded by $m_1m_1 = m_1$ and $m_1m_0 = 0$. Their equivalence can be verified by the following formula:

$$\vdash_{\text{NKA}} m_1m_1 = m_1 \wedge m_1m_0 = 0 \longrightarrow (m_0p)^*m_1 = (m_0p(m_0p + m_1 \cdot 1))^*m_1. \quad (5.1.1)$$

NKA Derivation: This formula can be derived in NKA by:

$$\begin{aligned}
&(m_0p(m_0p + m_1 \cdot 1))^*m_1 \\
&= (m_0pm_0p + m_0pm_1)^*m_1 && \text{(distributive-law)} \\
&= (m_0pm_0p)^*(m_0pm_1(m_0pm_0p)^*)^*m_1 && \text{(denesting)} \\
&= (m_0pm_0p)^*(m_0pm_1(1 + m_0pm_0p(m_0pm_0p)^*))^*m_1 && \text{(fixed-point)} \\
&= (m_0pm_0p)^*(m_0pm_1)^*m_1 && (m_1m_0 = 0) \\
&= (m_0pm_0p)^*(1 + m_0pm_1(1 + m_0pm_1(m_0pm_1)^*))m_1 && \text{(fixed-point)} \\
&= (m_0pm_0p)^*(1 + m_0pm_1)m_1 && (m_1m_0 = 0) \\
&= (m_0pm_0p)^*(1 + m_0p)m_1 && (m_1m_1 = m_1, \text{ distributive-law}) \\
&= (m_0p)^*m_1. && \text{(unrolling)}
\end{aligned}$$

By Theorem 1.1, we have $[\![\textsc{Unrolling1}]\!] = [\![\textsc{Unrolling2}]\!]$.

5.2 Loop Boundary

This rule is quantum-specific because it makes use of the reversible property of quantum operations. Consider programs Boundary1 and Boundary2 in Figure 4, where P is an arbitrary program. Here the unitary U acting on q does not affect the measurement on qubit w. In other words, quantum measurement M_0 and M_1 commute with U.

Program Encoding: We encode these program by expressions $\mathrm{Enc}(\textsc{Boundary1}) = (m_0upu^{-1})^*m_1$ and $\mathrm{Enc}(\textsc{Boundary2}) = u(m_0p)^*m_1u^{-1}$, where the encoder setting E can be inferred.

Condition Formulation: The reversibility property $UU^{-1} = U^{-1}U = I$ can be encoded by $uu^{-1} = u^{-1}u = 1$ (at the level of quantum superoperators). Besides, the commutativity property of measurement and U is encoded as $um_0 = m_0u$ and $um_1 = m_1u$. Then the formula we need to derive is

```
Unrolling1 ≡                    Unrolling2 ≡
while $M[q] = 0$ do             while $M[q] = 0$ do
    $P$                             $P$;  if $M[q] = 0$ then $P$
done.                           done.

Boundary1 ≡                     Boundary2 ≡
while $M[w] = 0$ do             $q := U[q]$;
    $q := U[q]$;                while $M[w] = 0$ do
    $P$;                            $P$;
    $q := U^{-1}[q]$            done;
done.                           $q := U^{-1}[q]$.
```

Figure 4. Two pairs of equivalent programs with conditions.

$$\vdash_{\text{NKA}} uu^{-1} = u^{-1}u = 1 \wedge um_0 = m_0u \wedge um_1 = m_1u \longrightarrow (m_0upu^{-1})^*m_1 = u(m_0p)^*m_1u^{-1}. \quad (5.2.1)$$

NKA Derivation: The derivation of this formula in NKA is

$$\begin{aligned}
&(m_0upu^{-1})^*m_1 \\
&= (um_0pu^{-1})^*m_1 && (um_0 = m_0u) \\
&= (1 + u(m_0pu^{-1}u)^*m_0pu^{-1})m_1 && \text{(product-star)} \\
&= u(m_0p)^*m_1u^{-1}. && (u^{-1}u = 1, \text{ fixed-point})
\end{aligned}$$

Then $[\![\textsc{Boundary1}]\!] = [\![\textsc{Boundary2}]\!]$ by Theorem 1.1.

Due to space limitations, we showcase in the appendix [47] the use of the Loop Boundary rule to optimize, as observed in [13], one leading quantum Hamiltonian simulation algorithm called quantum signal processing (QSP) [42], as well as its algebraic verification.

6 Normal Form of Quantum Programs

Here we use NKA to prove a quantum counterpart of the classic Böhm-Jacopini theorem [8], namely, a normal form of quantum **while** programs consisting of only a single loop. The normal form of classical programs depends on the folk operation, which copies the value of a variable to a new variable. However, in quantum programs, the no-cloning theorem prevents us from directly copying unknown states. Our approach is to store every measurement result in an augmented classical space and depends on the classical variables to manipulate the control flow of the program. We note a quantum version of the Böhm-Jacopini theorem was recently shown in [67], however, using a completely different and non-algebraic approach.

Let us illustrate our idea with a simple example below first. To unify the two **while** loops of Original into one, we redesign the control flow as in Constructed with a fresh classical guard variable $g \in \{0,1,2\}$.

```
ORIGINAL ≡
while $M_1[p] = 1$ do $P_1$ done;
while $M_2[p] = 1$ do $P_2$ done;
$g := |0\rangle$.
```

```
CONSTRUCTED ≡
$g := |1\rangle$;
while Meas$[g] > 0$ do
    if Meas$[g] > 1$ then
        if $M_2[p] = 1$ then $P_2$ else $g := |0\rangle$
    else
        if $M_1[p] = 1$ then $P_1$ else $g := |2\rangle$
done.
```

Here Meas$[g]$ is the computational basis measurement on variable g. When g is classical, Meas$[g]$ returns the value of g, and does not modify g. The variable g is used to store the measurement results and decide which branch the program executes in the next round. We prove $[\![\text{Original}]\!] = [\![\text{Constructed}]\!]$ via NKA, using the outline in Section 5.

Program Encoding: We encode $g := |i\rangle$ as g^i, and Meas$[g] > i$ as $g_{>i}$ and $g_{\le i}$. Then the two programs are encoded as

$$\text{Enc}(\text{Original}) = (m_{11}p_1)^* m_{10}(m_{21}p_2)^* m_{20} g^0,$$

$$\text{Enc}(\text{Constructed}) = g^1(g_{>0}(g_{>1}(m_{21}p_2 + m_{20}g^0) + g_{\le 1}(m_{11}p_1 + m_{10}g^2)))^* g_{\le 0}.$$

Condition Formulation: Since g is fresh, operations on g commutes with the quantum measurements M_1, M_2 and subprograms P_1, P_2. This is encoded as $g^i m_{jk} = m_{jk} g^i, g^i p_j = p_j g^i$. Two consecutive assignment on g will make the first one be overwritten, which is encoded as $g^i g^j = g^j$. On top of these, $g^i g_{>j} = g^i$ if $i > j$ and $g^i g_{>j} = 0$ if $i \le j$. Similarly, $g^i g_{\le j} = g^i$ if $i \le j$ and $g^i g_{\le j} = 0$ if $i > j$.

NKA derivation: To simplify the proof, let

$$X = g_{>0} g_{>1}(m_{21}p_2 + m_{20}g^0), \ Y = g_{>0} g_{\le 1}(m_{11}p_1 + m_{10}g^2).$$

Then Enc(Constructed) is equivalent to $g^1(X+Y)^* g_{\le 0}$. We simplify $g^i X^*$ first.

$$\begin{aligned} g^1 X^* &= g^1(1 + g_{>0}g_{>1}(m_{20}g^0 + m_{21}p_2)X^*) && \text{(fixed-point)} \\ &= g^1, && \text{(distributive-law)} \\ g^2 X^* &= g^2(g_{>0}g_{>1}m_{21}p_2)^*(g_{>0}g_{>1}m_{20}g^0 \\ &\quad \cdot (1 + g_{>0}g_{>1}m_{21}p_2(g_{>0}g_{>1}m_{21}p_2)^*))^* && \text{(denesting, fixed-point)} \\ &= (m_{21}p_2)^* g^2 (g_{>0}g_{>1}m_{20}g^0)^* && \text{(star-rewrite)} \\ &= (m_{21}p_2)^* g^2 (1 + g_{>0}g_{>1}m_{20}g^0) && \text{(fixed-point)} \\ &= (m_{21}p_2)^* (g^2 + m_{20}g^0). && \text{(distributive-law)} \end{aligned}$$

Consider $g^1(X+Y)^* = g^1 X^*(YX^*)^* = g^1(YX^*)^*$, and then

$$\begin{aligned} g^1(YX^*)^* &= g^1(g_{>0}g_{\le 1}m_{11}p_1X^*)^* \\ &\quad \cdot (g_{>0}g_{\le 1}m_{10}g^2X^*(g_{>0}g_{\le 1}m_{11}p_1X^*)^*)^* && \text{(denesting)} \\ &= (m_{11}p_1)^* g^1 (g_{>0}g_{\le 1}m_{10}m_{21}^*(g^2 + m_{20}g^0) \\ &\quad \cdot (1 + (g_{>0}g_{\le 1}m_{11}p_1X^*)(g_{>0}g_{\le 1}m_{11}p_1X^*)^*)) && \text{(star-rewrite, fixed-point)} \\ &= (m_{11}p_1)^* m_{10}(m_{21}p_2)^*(g^2 + m_{20}g^0). \end{aligned}$$

Insert the above equation into $g^1(X+Y)^* g_{\le 0}$.

$$\begin{aligned} g^1(X+Y)^* g_{\le 0} &= (m_{11}p_1)^* m_{10}(m_{21}p_2)^*(g^2 + m_{20}g^0) g_{\le 0} \\ &= (m_{11}p_1)^* m_{10}(m_{21}p_2)^* m_{20}g^0. \end{aligned}$$

This is exactly Enc(Constructed) = Enc(Original). Theorem 1.1 gives $[\![\text{Constructed}]\!] = [\![\text{Original}]\!]$. Hence the two loops have been merged into one, with an additional classical space which is restored to 0 at the end.

We employ a similar idea to arbitrary programs by induction. Note that our above example corresponds to the case $S_1; S_2$ in induction. And our analysis above, which results in an equivalent program with one **while**-loop and additional classical space, constitutes a proof in that case. The more complicated cases are proved similarly, whose details are in the appendix [47].

Theorem 6.1. *For any quantum while program P over Hilbert space $\mathcal{H}$, there are a classical space C and a quantum while program*

$$P_0; \ \textbf{while } M \textbf{ do } P_1 \textbf{ done}; \ p_C := |0\rangle \qquad (6.0.1)$$

equivalent to P; $p_C := |0\rangle$ over $\mathcal{H} \otimes C$, where P_0, P_1 are while-free, $p_C := |0\rangle$ resets the classical variables in C to $|0\rangle$.

7 Non-idempotent Kleene Algebra with Tests

As we stated before, NKA is not specifically designed for quantum programs: the measurements are treated as normal processes. Further characterization of measurements will grant finer algebraic structure. KAT introduces tests into KA, relying on the ability to simultaneously represent branching and predicates by Boolean algebra. However, for quantum programs, there is a gap between branching and predicates, which requires us to treat predicates and branching separately. We introduce effect algebra as a subalgebra of NKA to tackle quantum predicates in Section 7.1. As for branching, quantum measurements are abstracted as algebraic rules based on predicates. These lead to non-idempotent Kleene algebra with tests (NKAT) in Section 7.2. As an application, we show how propositional quantum Hoare logic is subsumed into algebraic rules of NKAT in Section 7.3 and Section 7.4.

7.1 Effect Algebra

The notion of quantum predicates was defined in [15] as an operator $A \in \mathcal{PO}(\mathcal{H})$ satisfying $\|A\| \le 1$, and its negation $\overline{A} = I_{\mathcal{H}} - A$. In the quantum foundations literature, quantum predicates are also called *effects*. Their algebraic properties have been extensively studied as effect algebras.

Definition 7.1 ([22])**.** *An effect algebra (EA) is a 4-tuple $(\mathcal{L}, \oplus, 0, e)$, where $0, e \in \mathcal{L}$, and $\oplus : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$ is a partial binary operation satisfying the following properties: for any $a, b, c \in \mathcal{L}$,*

1. *if $a \oplus b$ is defined then $b \oplus a$ is defined and $a \oplus b = b \oplus a$;*
2. *if $a \oplus b$ and $(a \oplus b) \oplus c$ are defined, then $b \oplus c$ and $a \oplus (b \oplus c)$ are defined and $(a \oplus b) \oplus c = a \oplus (b \oplus c)$;*
3. *if $a \oplus e$ is defined, then $a = 0$;*
4. *for every $a \in \mathcal{L}$ there exists a unique $\overline{a} \in \mathcal{L}$ such that $a \oplus \overline{a} = e$;*
5. *for every $a \in \mathcal{L}$, $0 \oplus a = a$.*

The fourth rule of the effect algebra defines a unary operator, the *negation* over $\mathcal{L}$, denoted by $\bar{a}$ for $a \in \mathcal{L}$.

An effect algebra is easily embedded in NKA by viewing $\oplus$ as a restricted $+$ of NKA. Then we need to identify the correspondence of predicates in the quantum path model.

Definition 7.2. *For a predicate A, a constant superoperator $C_A \in \mathcal{QC}(\mathcal{H})$ for $A \in \mathcal{PO}(\mathcal{H})$ is defined by*

$$C_A(\rho) = \mathrm{tr}[\rho]A. \tag{7.1.1}$$

We let $\mathcal{P}_{\mathrm{Pred}}(\mathcal{H}) = \{\langle C_A \rangle^{\uparrow} : A \in \mathcal{PO}(\mathcal{H}), \|A\| \le 1\}$ be the subset of $\mathcal{P}(\mathcal{H})$ containing the lifted constant superoperator.

A partial binary addition $\oplus$ over $\mathcal{P}_{\mathrm{Pred}}(\mathcal{H})$ inherits from the addition in $\mathcal{P}(\mathcal{H})$, defined by:

$$\langle C_A \rangle^{\uparrow} \oplus \langle C_B \rangle^{\uparrow} = \begin{cases} \langle C_A \rangle^{\uparrow} + \langle C_B \rangle^{\uparrow} & \langle C_A \rangle^{\uparrow} + \langle C_B \rangle^{\uparrow} \le \langle C_{I_{\mathcal{H}}} \rangle^{\uparrow}, \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

Lemma 7.3. *$(\mathcal{P}_{\mathrm{Pred}}(\mathcal{H}), \oplus, \mathcal{O}_{\mathcal{H}}, \langle C_{I_{\mathcal{H}}} \rangle^{\uparrow})$ forms an effect algebra. Specifically, the negation of it satisfies $\overline{\langle C_A \rangle^{\uparrow}} = \langle C_{\overline{A}} \rangle^{\uparrow}$.*

The proof is straightforward and in the appendix [47].

7.2 Non-idempotent Kleene Algebras with Tests

We can characterize quantum measurements with the help of predicates, for which we propose *partitions* algebraically.

Definition 7.4. *An* NKAT *is a many-sorted algebra $(\mathcal{K}, \mathcal{L}, \mathcal{N}, +, \cdot, *, \le, 0, 1, e)$ such that*

1. *$(\mathcal{K}, +, \cdot, *, \le, 0, 1)$ is an* NKA*;*
2. *$\mathcal{L}$ is a subset of $\mathcal{K}$, and $(\mathcal{L}, \oplus, 0, e)$ is an effect algebra, where $\oplus$ is the restriction of $+$ w.r.t. top element e and partial order $\le$; that is, for any $a, b \in \mathcal{L}$*

$$a \oplus b = \begin{cases} a + b & a + b \le e, \\ \textit{undefined} & \textit{otherwise;} \end{cases} \tag{7.2.1}$$

3. *$\mathcal{N}$ is a set of tuples $(m_i)_{i \in I}$, where I are finite index sets and $m_i \in \mathcal{K}$, satisfying:*
   a. *each entry in the tuples satisfies $m_i\mathcal{L} \subseteq \mathcal{L}$; that is, for $a \in \mathcal{L}$, $m_i a \in \mathcal{L}$.*
   b. *for each tuple, $\sum_{i \in I} m_i e = e$.*

The tuples in $\mathcal{N}$ are called partitions.

We use $\mathcal{L}$ to characterize quantum predicates, and $\mathcal{N}$ to characterize branching in quantum programs. For a quantum measurement $\{M_i\}_{i \in I}$, its dual superoperators transform quantum predicates to quantum predicates: $\mathcal{E}^{\dagger}_{M_i}(A) = M_i^{\dagger} A M_i$. This is captured by $m_i\mathcal{L} \subseteq \mathcal{L}$. Besides, general quantum measurements satisfies $\sum_{i \in I} M_i^{\dagger} M_i = I$, which is captured by $\sum_{i \in I} m_i e = e$, since e represents predicate $I_{\mathcal{H}}$. [4]

Definition 7.5. *The set of quantum measurements lifted as quantum path actions in the dual sense is $\mathcal{P}_{\mathrm{Meas}}(\mathcal{H}) = \left\{ \left(\langle \mathcal{M}_i^{\dagger} \rangle^{\uparrow} \right)_{i \in I} : \mathcal{M}_i(\rho) = M_i \rho M_i^{\dagger}, \sum_{i \in I} M_i^{\dagger} M_i = I_{\mathcal{H}} \right\}$.*

Then we augment the quantum path model in the NKAT framework, supporting quantum predicates ($\mathcal{P}_{\mathrm{Pred}}(\mathcal{H})$) and quantum measurements ($\mathcal{P}_{\mathrm{Meas}}(\mathcal{H})$).

Theorem 7.6. *The NKAT axioms are sound for the algebra $(\mathcal{P}(\mathcal{H}), \mathcal{P}_{\mathrm{Pred}}(\mathcal{H}), \mathcal{P}_{\mathrm{Meas}}(\mathcal{H}), +, \diamond, *, \le, \mathcal{O}_{\mathcal{H}}, \mathcal{I}_{\mathcal{H}}, \langle C_{I_{\mathcal{H}}} \rangle^{\uparrow})$.*

Note we have substituted the right composition operation $\diamond$ for the left composition operation ; in $\mathcal{P}(\mathcal{H})$. This is mainly because our interpretation now uses dual superoperators. The verification of each axiom is standard. The detailed proofs are included in the appendix [47].

Several useful rules are derivable in NKAT, and their proofs are in the appendix [47].

Lemma 7.7. *The following formulae are derivable in NKAT. Here I is a finite index set, a, b, a_i are elements of the effect subalgebra, $(m_i)_{i \in I}$ is a partition.*

1. *$0 \le a \le e$;* 2. *$a + \overline{a} = e$;* 3. *$\overline{\overline{a}} = a$;*
4. *$a \le b \to \overline{b} \le \overline{a}$;* *(negation-reverse)*
5. *$\overline{\sum_{i \in I} m_i a_i} = \sum_{i \in I} m_i \overline{a_i}$.* *(partition-transform)*

7.3 Encoding of Quantum Hoare Triples

A natural usage of classical predicates is reasoning via Hoare triples. With an algebraic representation of quantum predicates and programs, we can encode quantum Hoare triples as algebraic formulae. A quantum Hoare triple is a judgment of the form $\{A\}P\{B\}$ where A, B are quantum predicates and P is a quantum program. It refers to partial correctness [64], denoted by $\models_{par} \{A\}P\{B\}$, if for all input $\rho \in \mathcal{D}(\mathcal{H})$ there is

$$\mathrm{tr}(A\rho) \le \mathrm{tr}(B \llbracket P \rrbracket (\rho)) + \mathrm{tr}(\rho) - \mathrm{tr}(\llbracket P \rrbracket (\rho)). \tag{7.3.1}$$

[4]Our characterization of measurements matches positive-operator-valued measurements (POVM), the most general quantum measurements. We can further classify structures inside $\mathcal{N}$ to depict specific classes of quantum measurements. For example, projection-valued measurements (PVM) can be modeled as tuples $(m_i)_{i \in I}$ where $m_i m_j = m_i$ if $i = j$, otherwise $m_i m_j = 0$.

Furthermore, a set of projective and pair-wise commutative measurement superoperators, defined by $C(\mathcal{H}) = \{\mathcal{E} \in \mathcal{QC}(\mathcal{H}) : \mathcal{E}(\rho) = D\rho D^{\dagger}, D \text{ is diagonal}, D^2 = D\}$, represents the measurement superoperators in probabilistic programs. A Boolean algebra can be observed from it. It would be an interesting future direction to investigate the algebraic relation between NKAT and this Boolean algebra.

$$
\begin{array}{llll}
\text{(Ax.UT)} & \{U^\dagger A U\}\ \overline{q} := U[\overline{q}]\ \{A\} & \text{(Ax.In)} & \left\{\sum_i |i\rangle_q\langle 0|A|0\rangle_q\langle i|\right\} q := |0\rangle\ \{A\} \\
\text{(Ax.Sk)} & \{A\}\ \textbf{skip}\ \{A\} & \text{(R.OR)} & \dfrac{A \sqsubseteq A' \quad \{A'\}P\{B'\} \quad B' \sqsubseteq B}{\{A\}P\{B\}} \\
\text{(Ax.Ab)} & \{I_{\mathcal{H}}\}\ \textbf{abort}\ \{O_{\mathcal{H}}\} & \text{(R.IF)} & \dfrac{\{A_i\}P_i\{B\} \text{ for all } i}{\{\sum_i \mathcal{M}_i^\dagger(A_i)\}\textbf{case } M \xrightarrow{i} P_i \textbf{ end}\{B\}} \\
\text{(R.SC)} & \dfrac{\{A\}P_1\{B\} \quad \{B\}P_2\{C\}}{\{A\}P_1;P_2\{C\}} & \text{(R.LP)} & \dfrac{\{B\}P\{C\} \quad C = \mathcal{M}_0^\dagger(A) + \mathcal{M}_1^\dagger(B)}{\{C\}\textbf{while } M = 1 \textbf{ do } P \textbf{ done}\{A\}}
\end{array}
$$

Figure 5. A proof system for partial correctness of quantum programs. Propositional quantum Hoare logic includes the rules marked red in this figure (the lower six rules).

Then partial correctness $\models_{par} \{A\}P\{B\}$ can be encoded as an inequality $p\overline{b} \leq \overline{a}$, where p is the encoding of program P, and effect algebra elements a, b are the encoding of constant superoperators C_A and C_B. This encoding can be interpreted by a dual interpretation $Q_{\text{int}}^\dagger$. [5] By setting any non-zero input for $Q_{\text{int}}^\dagger(p\overline{b}) \leq Q_{\text{int}}^\dagger(\overline{a})$ and Lemma 3.8.(ii), it turns to $[\![P]\!]^\dagger(I - B) \sqsubseteq I - A$, which is equivalent to $\models_{par} \{A\}P\{B\}$.

7.4 Propositional Quantum Hoare Logic

An important feature of KAT is that KAT subsumes the deductive system of propositional Hoare logic, which contains the rules directly related to the control flow of classical while programs but not the rule for assignments [34]. As a counterpart, quantum Hoare logic is an important tool in the verification and analysis of quantum programs. A sound and (relatively) complete proof system for partial correctness of quantum **while** programs presented in Figure 5 is discussed in [63]. We aim to subsume in NKAT a fragment of quantum Hoare logic, called *propositional* quantum Hoare logic.

Due to the no-cloning of quantum information, the role of assignment is played by initialization and unitary transformation together in quantum programming. In quantum Hoare logic, the rule (Ax.In) and (Ax.UT) for them include atomic transformations, which cannot be captured by algebraic methods. As such, propositional quantum Hoare logic will treat these rules as atomic propositions and work with the following program syntax

$$
\begin{aligned}
P \quad ::= \ & p \mid \textbf{skip} \mid \textbf{abort} \mid P_1;P_2 \mid \\
& \textbf{case } M[\overline{q}] \xrightarrow{i} P_i \textbf{ end} \mid \textbf{while } M[\overline{q}] = 1 \textbf{ do } P_1 \textbf{ done}.
\end{aligned}
$$

Therefore, the deductive system of propositional quantum Hoare logic consists of the rules marked red in Figure 5. Its relative completeness and soundness can be proved similarly to the original quantum Hoare logic [63] as a routine exercise.

[5]A *dual interpretation* $Q_{\text{int}}^\dagger$ is defined similar to Q_{int} except for $Q_{\text{int}}^\dagger(e \cdot f) = Q_{\text{int}}^\dagger(e) \diamond Q_{\text{int}}^\dagger(f)$ and $Q_{\text{int}}^\dagger(a) = \langle \text{eval}(a)^\dagger \rangle^\uparrow$. It describes the dual superoperators lifted to $\mathcal{P}(\mathcal{H})$. Properties of Q_{int} like Theorem 4.2, Corollary 4.3 hold for the dual interpretation similarly. Analogous of Theorem 4.5, $Q_{\text{int}}^\dagger(\text{Enc}(P)) = \langle [\![P]\!]^\dagger \rangle^\uparrow$, holds as well.

By the discussions in Section 7.3, the partial correctness of quantum Hoare triples can be encoded in NKAT. For a quantum measurement $\{M_i\}_{i\in I}$ we have an additional normalization rule $\sum_i M_i^\dagger M_i = I$, which is encoded as $\sum_i m_i e = e$. Then the encoding of these rules is

$$
\begin{cases}
\text{(Ax.Sk)}: & 1\overline{a} \leq \overline{a}, \\
\text{(Ax.Ab)}: & 0\overline{0} \leq \overline{1}, \\
\text{(R.OR)}: & a \leq a' \wedge p\overline{b'} \leq \overline{a'} \wedge b' \leq b \rightarrow p\overline{b} \leq \overline{a}, \\
\text{(R.IF)}: & \left(\bigwedge_{i\in I} p_i\overline{b} \leq \overline{a_i}\right) \rightarrow (\sum_{i\in I} m_i p_i)\overline{b} \leq \overline{\sum_i m_i a_i}, \\
\text{(R.SC)}: & p_1\overline{b} \leq \overline{a} \wedge p_2\overline{c} \leq \overline{b} \rightarrow p_1p_2\overline{c} \leq \overline{a}, \\
\text{(R.LP)}: & p\overline{m_0a + m_1b} \leq \overline{b} \rightarrow (m_1p)^* m_0\overline{a} \leq \overline{m_0a + m_1b}.
\end{cases}
$$

Here I is a finite index set, $p, p_i \in \mathcal{K}$, elements $a, b, c, a', b', a_i \in \mathcal{L}$, and $(m_i)_{i\in I}$ are partitions.

Theorem 7.8. *With partitions $(m_i)_{i\in I}$, the formulae above are derivable in NKAT.*

Proof.

1. (Ax.Sk): $1\overline{a} = \overline{a}$.
2. (Ax.Ab): $0\overline{0} = 0 \leq \overline{1}$ by positivity.
3. (R.OR): By negation-reverse, we have $\overline{a'} \leq \overline{a}$ and $\overline{b} \leq \overline{b'}$. So $p\overline{b} \leq p\overline{b'} \leq \overline{a'} \leq \overline{a}$.
4. (R.IF): Applying partition-transform, $(\sum_{i\in I} m_i p_i)\overline{b} = \sum_{i\in I} m_i p_i \overline{b} \leq \sum_{i\in I} m_i \overline{a_i} = \overline{\sum_i m_i a_i}$.
5. (R.SC): $p_1(p_2\overline{c}) \leq p_1\overline{b} \leq \overline{a}$.
6. (R.LP): By partition-transform, $\overline{m_0a + m_1b} = m_0\overline{a} + m_1\overline{b}$. With $p\overline{m_0a + m_1b} \leq \overline{b}$, we have
$$m_0\overline{a} + m_1p\overline{m_0a + m_1b} \leq m_0\overline{a} + m_1\overline{b} = \overline{m_0a + m_1b}.$$
Then $(m_1p)^* m_0\overline{a} \leq \overline{m_0a + m_1b}$ is concluded by applying $q + pr \leq r \rightarrow p^* q \leq r$.

□

It is clear that the NKAT subsumes the encoding of propositional quantum Hoare logic.

Acknowledgement

We thank the anonymous reviewers and our shepherd, Calvin Smith, for their helpful feedback. Y.P. and X.W. were supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Testbed Pathfinder Program under Award Number DE-SC0019040 and the U.S. Air Force Office of Scientific Research MURI grant FA9550-16-1-0082. M.Y. was supported by the National Key R&D Program of China (2018YFA0306701) and the National Natural Science Foundation of China (61832015).

References

[1] Ali Javadi Abhari, Arvin Faruque, Mohammad Javad Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, Fred Chong, Margaret Martonosi, Martin Suchara,

Ken Brown, Massoud Pedram, and Todd Brun. 2012. *Scaffold: Quantum Programming Language.* Technical Report TR-934-12. Princeton University.

[2] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, et al. 2019. Qiskit: An Open-source Framework for Quantum Computing.

[3] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. In *Proc. 41st ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages (POPL'14)*. ACM, San Diego, California, USA, 113–126.

[4] Allegra Angus and Dexter Kozen. 2001. *Kleene Algebra with Tests and Program Schematology.* Technical Report TR2001-1844. Computer Science Department, Cornell University.

[5] Alexandru Baltag and Sonja Smets. 2011. Quantum logic as a dynamic logic. *Synthese* 179, 2 (2011), 285–306.

[6] Gilles Barthe, Justin Hsu, Mingsheng Ying, Nengkun Yu, and Li Zhou. 2019. Relational Proofs for Quantum Programs. *Proc. ACM Program. Lang.* 4, POPL, Article 21 (Dec. 2019), 29 pages. https://doi.org/10.1145/3371089

[7] Stephen L Bloom and Zoltán Ésik. 2009. Axiomatizing rational power series over natural numbers. *Information and Computation* 207, 7 (2009), 793–811.

[8] Corrado Böhm and Giuseppe Jacopini. 1966. Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules. *Commun. ACM* 9, 5 (May 1966), 366–371. https://doi.org/10.1145/355592.365646

[9] Filippo Bonchi, Marcello Bonsangue, Michele Boreale, Jan Rutten, and Alexandra Silva. 2012. A coalgebraic perspective on linear weighted automata. *Information and Computation* 211 (2012), 77 – 105. https://doi.org/10.1016/j.ic.2011.12.002

[10] Olivier Brunet and Philippe Jorrand. 2004. Dynamic Quantum Logic for Quantum Programs. *International Journal of Quantum Information* 2, 1 (2004).

[11] Angela Sara Cacciapuoti, Marcello Caleffi, Francesco Tafuri, Francesco Saverio Cataliotti, Stefano Gherardini, and Giuseppe Bianchi. 2020. Quantum Internet: Networking Challenges in Distributed Quantum Computing. *IEEE Network* 34, 1 (2020), 137–143. https://doi.org/10.1109/MNET.001.1900092

[12] Rohit Chadha, Paulo Mateus, and Amílcar Sernadas. 2006. Reasoning About Imperative Quantum Programs. *Electronic Notes in Theoretical Computer Science* 158 (2006).

[13] Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. 2018. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences* 115, 38 (2018), 9456–9461.

[14] Ernie Cohen, Dexter Kozen, and Frederick Smith. 1996. *The complexity of Kleene algebra with tests.* Technical Report TR96-1598. Computer Science Department, Cornell University.

[15] Ellie D'Hondt and Prakash Panangaden. 2006. Quantum Weakest Preconditions. *Mathematical Structures in Computer Science* 16, 3 (2006).

[16] Samuel Eilenberg. 1974. *Automata, languages, and machines.* Academic press.

[17] Zoltán Ésik and Werner Kuich. 2004. Inductive *-semirings. *Theoretical Computer Science* 324, 1 (2004), 3–33.

[18] Yuan Feng, Runyao Duan, Zhengfeng Ji, and Mingsheng Ying. 2007. Proof Rules for the Correctness of Quantum Programs. *Theoretical Computer Science* 386, 1-2 (2007).

[19] Michael J. Fischer and Richard E. Ladner. 1979. Propositional dynamic logic of regular programs. *J. Comput. System Sci.* 18, 2 (1979), 194 – 211. https://doi.org/10.1016/0022-0000(79)90046-1

[20] Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. 2016. Probabilistic NetKAT. In *25th European Symposium on Programming (ESOP 2016) (Lecture Notes in Computer Science, Vol. 9632)*, Peter Thiemann (Ed.). Springer, Eindhoven, The Netherlands, 282–309. https://doi.org/10.1007/978-3-662-49498-1_12

[21] Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. 2015. A Coalgebraic Decision Procedure for NetKAT. In *Proc. 42nd ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages (POPL'15)*. ACM, Mumbai, India, 343–355.

[22] David J Foulis and Mary K Bennett. 1994. Effect algebras and unsharp quantum logics. *Foundations of physics* 24, 10 (1994), 1331–1352.

[23] Simon J. Gay. 2006. Quantum Programming Languages: Survey and Bibliography. *Mathematical Structures in Computer Science* 16, 4 (2006).

[24] Google. 2018. https://github.com/quantumlib/Cirq.

[25] Jonathan Grattage. 2005. A Functional Quantum Programming Language. In *LICS*.

[26] Alexander S Green, Peter LeFanu Lumsdaine, Neil J Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: a scalable quantum programming language. In *PLDI*. 333–342.

[27] Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu, and Michael Hicks. 2019. A verified optimizer for quantum circuits. *arXiv preprint arXiv:1912.02250* (2019).

[28] C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (Oct. 1969), 576–580.

[29] Yoshihiko Kakutani. 2009. A Logic for Formal Verification of Quantum Programs. In *ASIAN 2009*. 79–93.

[30] Stefan Kiefer, Andrzej Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. 2013. On the complexity of equivalence and minimisation for Q-weighted automata. *arXiv preprint arXiv:1302.2818* (2013).

[31] S. C. Kleene. 1956. *Representation of Events in Nerve Nets and Finite Automata.* Princeton University Press, Princeton, 3 – 42. https://doi.org/10.1515/9781400882618-002

[32] Dexter Kozen. 1990. *A completeness theorem for Kleene algebras and the algebra of regular events.* Technical Report. Cornell University.

[33] Dexter Kozen. 1997. Kleene algebra with tests. *ACM Trans. Programming Languages and Systems (TOPLAS)* 19, 3 (May 1997), 427–443. https://doi.org/10.1145/256167.256195

[34] Dexter Kozen. 2000. On Hoare logic and Kleene algebra with tests. *Trans. Computational Logic* 1, 1 (July 2000), 60–76.

[35] Dexter Kozen. 2017. On the Coalgebraic Theory of Kleene Algebra with Tests. In *Rohit Parikh on Logic, Language and Society*, Can Başkent, Lawrence S. Moss, and Ramaswamy Ramanujam (Eds.). Outstanding Contributions to Logic, Vol. 11. Springer, 279–298.

[36] Dexter Kozen and Maria-Cristina Patron. 2000. Certification of compiler optimizations using Kleene algebra with tests. In *Proc. 1st Int. Conf. Computational Logic (CL2000)* (London) *(Lecture Notes in Artificial Intelligence, Vol. 1861)*, John Lloyd, Veronica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luis Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey (Eds.). Springer-Verlag, London, 568–582.

[37] Dexter Kozen and Frederick Smith. 1996. Kleene algebra with tests: Completeness and decidability. In *Proc. 10th Int. Workshop Computer Science Logic (CSL'96) (Lecture Notes in Computer Science, Vol. 1258)*, D. van Dalen and M. Bezem (Eds.). Springer-Verlag, Utrecht, The Netherlands, 244–259.

[38] Wojciech Kozlowski and Stephanie Wehner. 2019. Towards Large-Scale Quantum Networks. In *Proceedings of the Sixth Annual ACM International Conference on Nanoscale Computing and Communication* (Dublin, Ireland) *(NANOCOM '19)*. Association for Computing Machinery, New York, NY, USA, Article 3, 7 pages. https://doi.org/10.1145/3345312.3345497

[39] Karl Kraus, Arno Böhm, John D Dollard, and WH Wootters. 1983. States, effects, and operations: fundamental notions of quantum theory. Lectures in mathematical physics at the University of Texas at Austin. *Lecture notes in physics* 190 (1983).

[40] Werner Kuich and Arto Salomaa. 1985. *Semirings, Automata, Languages.* Springer-Verlag, Berlin, Heidelberg.

[41] Yangjia Li and Mingsheng Ying. 2017. Algorithmic Analysis of Termination Problems for Quantum Programs. 2, POPL, Article 35 (Dec. 2017), 29 pages.

[42] Guang Hao Low and Isaac L Chuang. 2017. Optimal Hamiltonian simulation by quantum signal processing. *Physical review letters* 118, 1 (2017), 010501.

[43] Mike Mislove. 2006. On Combining Probability and Nondeterminism. *Electronic Notes in Theoretical Computer Science* 162 (2006), 261 – 265. Proceedings of the Workshop Essays on Algebraic Process Calculi (APC 25).

[44] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

[45] Bernhard Ömer. 2003. *Structured Quantum Programming.* Ph. D. Dissertation. Vienna University of Technology.

[46] Jennifer Paykin, Robert Rand, and Steve Zdancewic. 2017. QWIRE: A Core Language for Quantum Circuits *(POPL 2017)*. 846–858.

[47] Yuxiang Peng, Mingsheng Ying, and Xiaodi Wu. 2021. Algebraic Reasoning of Quantum Programs via Non-Idempotent Kleene Algebra. *arXiv preprint arXiv:2110.07018* (2021).

[48] Damien Pous. 2015. Symbolic Algorithms for Language Equivalence and Kleene Algebra with Tests. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Mumbai, India) *(POPL '15)*. Association for Computing Machinery, New York, NY, USA, 357–368. https://doi.org/10.1145/2676726.2677007

[49] Rigetti. 2018. https://www.rigetti.com/forest.

[50] Amr Sabry. 2003. Modeling Quantum Computing in Haskell. In *The Haskell Workshop*.

[51] Jeff W. Sanders and Paolo Zuliani. 2000. Quantum Programming. In *MPC*.

[52] Peter Selinger. 2004. A Brief Survey of Quantum Programming Languages. In *FLOPS*.

[53] Peter Selinger. 2004. Towards a Quantum Programming Language. *Mathematical Structures in Computer Science* 14, 4 (2004).

[54] Alexandra Silva. 2010. *Kleene coalgebra.* Ph. D. Dissertation. Radboud University Nijmegen.

[55] Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. 2019. Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time. *Proc. ACM Program. Lang.* 4, POPL, Article 61 (Dec. 2019), 28 pages. https://doi.org/10.1145/3371129

[56] Sam Staton. 2015. Algebraic Effects, Linearity, and Quantum Programming Languages. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Mumbai, India) *(POPL '15)*. Association for Computing Machinery, New York, NY, USA, 395–406. https://doi.org/10.1145/2676726.2676999

[57] Larry J Stockmeyer and Albert R Meyer. 1973. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*. 1–9.

[58] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL. In *RWDSL*.

[59] Dominique Unruh. 2019. Quantum Relational Hoare Logic. 3, POPL, Article 33 (Jan. 2019), 31 pages.

[60] Daniele Varacca and Glynn Winskel. 2006. Distributing probability over non-determinism. *Mathematical Structures in Computer Science* 16, 1 (2006), 87–113. https://doi.org/10.1017/S0960129505005074

[61] John Watrous. 2018. *The Theory of Quantum Information.* Cambridge University Press. https://doi.org/10.1017/9781316848142

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

[63] Mingsheng Ying. 2011. Floyd–Hoare Logic for Quantum Programs. *ACM Transactions on Programming Languages and Systems* 33, 6 (2011).

[64] Mingsheng Ying. 2016. *Foundations of Quantum Programming*. Morgan Kaufmann.

[65] Mingsheng Ying. 2019. Toward automatic verification of quantum programs. *Formal Aspects of Computing* 31, 1 (01 Feb 2019), 3–25.

[66] Mingsheng Ying, Shenggang Ying, and Xiaodi Wu. 2017. Invariants of Quantum Programs: Characterisations and Generation *(POPL 2017)*. 818–832.

[67] Nengkun Yu. 2019. Quantum Temporal Logic. *arXiv e-prints*, Article arXiv:1908.00158 (July 2019), arXiv:1908.00158 pages. arXiv:1908.00158 [cs.LO]

[68] Nengkun Yu and Jens Palsberg. 2021. Quantum Abstract Interpretation. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (Virtual, Canada) *(PLDI 2021)*. Association for Computing Machinery, New York, NY, USA, 542–558. https://doi.org/10.1145/3453483.3454061

[69] Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An Applied Quantum Hoare Logic *(PLDI 2019)*. 1149–1162.