Quantum Expectation Transformers for Cost Analysis*

Martin Avanzini
Inria Sophia Antipolis - Méditerranée
France

Georg Moser
Department of Computer Science
Universität Innsbruck
Innsbruck, Austria

Romain Péchoux
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy
France

Simon Perdrix
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy
France

Vladimir Zamdzhiev
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy
France

1 Introduction

Quantum computation is a promising and emerging computational paradigm which can efficiently solve problems considered to be intractable on classical computers [4, 12]. However, the unintuitive nature of quantum mechanics poses interesting and challenging questions for the design and analysis of quantum programming languages. Indeed, the quantum program dynamics are considerably more complicated compared to the behaviour of classical probabilistic programs. Therefore, formal reasoning about quantum programs requires the development of novel methods and tools.

An important open problem is to compute the expected resource usage of quantum programs. For example, this may be used to determine: (1) the expected runtime; (2) the expected number of quantum gates; or (3) the amount of quantum resources (in an application-specific sense) required by quantum programs, etc. The difficulty of this problem, which is undecidable, requires using elaborate methods to solve it whenever possible. These methods for estimating resource usage must be compositional, systematic, and, preferably, tractable; this excludes de facto any direct use of the operational semantics.

We address this open problem by establishing a weakest precondition reasoning in the form of a *quantum expectation transformer*, named `qet`-*calculus*, that is rich enough to recover earlier wp-calculi in the context of classical programs as well as denotational semantics for quantum programs. Further, the calculus appears to be the right foundation for subsequent automation of the method, which however, is left for future work. The exact solution of the expected cost problem can be recovered via this calculus, and furthermore, our method may also be used to find *approximate* solutions by identifying suitable upper bounds. Therefore, our method provides a basis for attacking and ameliorating this undecidable problem in a systematic and compositional way.

Our Contributions. As a first step towards achieving our main objective, we introduce a new domain-theoretic notion, called a *cost structure* (Section 2). It is based on Kegelspitzen [7], which are dcpo's (directed-complete partial orders) equipped with a suitable convex structure that may be used to reason about the semantics of probabilistic [6, 10] and quantum programming languages [5]. A cost structure is then a pair $(\mathsf{S}, \hat{+})$ of a Kegelspitze S together with a *cost addition* operation $\hat{+}$ that allows us to model resource consumption in a coherent way.

We introduce a mixed classical-quantum programming language on which we formally define the expected cost and the expected value of programs. Our programming language (Section 3) supports conditional branching, while loops, the usual quantum primitives (including quantum measurements), classical data, and a special statement for resource consumption. To seamlessly model the combination of cost primitives and probabilistic choice — induced by quantum measurements — we define the operational semantics of our language as a probabilistic abstract reduction system [3], whose reduction rules are annotated by costs [2].

In Section 4, we introduce the aformentioned qet-calculus, which can be seen as a generalisation of previous work on predicate transformers and probabilistic expectation transformers. For a given cost structure $(\mathsf{S}, \hat{+})$, our quantum expectation transformer is a semantic function

$$\mathtt{qet}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to \mathsf{S}^{\mathsf{State}} \to \mathsf{S}^{\mathsf{State}}$$

which maps programs and *expectations* (functions from quantum program states to a cost structure) to expectations, in a continuation passing style. We prove that our semantics enjoys nice algebraic and domain-theoretic properties and that it is sound and adequate with respect to the operational semantics. As a consequence, we prove that the expected cost of a program in our mixed classical-quantum language (as defined via the operational semantics) is precisely recovered by using our quantum expectation transformer (Corollary 4.4). Furthermore, because our semantics is defined in a suitable level of generality, by choosing an appropriate cost structure $(\mathsf{S}, \hat{+})$, we show how a strongly adequate quantum denotational semantics may be defined as a *special case* (§4.1), which highlights important connections between our approach and denotational semantics of probabilistic and quantum programming languages.

*Extended abstract for the paper https://arxiv.org/abs/2201.09361 which will appear in LICS'22.

The usefulness of our methods are illustrated through a running example that performs (unbounded) coin tossing using quantum resources and through more involved quantum programs in the full paper (omitted here).

2 Kegelspitzen and Cost Structures

We begin by defining a notion of *cost structure* based on the domain-theoretic and convex structure of Kegelspitzen. This is used in later sections by our quantum expectation transformers in order to formalise the semantics.

Kegelspitzen [7] are cpo's (complete partial orders) that enjoy a convex structure.

Definition 2.1. A Kegelspitze is a pointed barycentric algebra K equipped with an ω-complete partial order such that, (1) scalar multiplication $(r, a) \mapsto r \cdot a \colon [0,1] \times K \to K$ is ω-continuous in both arguments, and (2) for every $r \in [0,1]$ convex combination $(a, b) \mapsto a +_r b \colon K \times K \to K$ is ω-continuous in both arguments.

Example 2.2. The real unit interval $[0,1]$ is a Kegelspitze in the usual order when we define $a +_r b \triangleq ra + (1-r)b$ and $\perp \triangleq 0$. The same assignment can also be used to equip the extended non-negative reals $\mathbb{R}^{+\infty} \triangleq \mathbb{R}^+ \cup \{\infty\}$ with the structure of a Kegelspitze. Note that the non-negative reals $\mathbb{R}^+$ is *not* a Kegelspitze, because it lacks an ω-cpo structure.

Example 2.3. A *density matrix* is a positive semi-definite hermitian matrix A, such that $\mathrm{tr}(A) = 1$. A *subdensity matrix* is a positive semi-definite hermitian matrix A, such that $\mathrm{tr}(A) \leq 1$. Let $D_n \subseteq \mathbb{C}^{n \times n}$ be the set of subdensity matrices of dimension n. Then D_n is an ω-cpo when equipped with the Löwner order: $A \leq B$ iff $B - A$ is positive semi-definite [11]. Moreover, D_n has the structure of a Kegelspitze under the assignment $\perp \triangleq \mathbf{0}$ and $A +_r B \triangleq rA + (1-r)B$.

In quantum programming semantics, we use *sub*density matrices in order to account for the probability of non-termination. Kegelspitzen may also be used to define (countable) convex sums.

We now formalize a notion of *cost structure* for expectation transformers in the context of quantum programs. This can be seen as a Kegelspitze equipped with an operation for injecting a cost — modeled as a positive real number — into the Kegelspitze, which satisfies some coherence conditions with respect to the structure of the Kegelspitze.

Definition 2.4. A *cost structure* $\mathcal{S} = (\mathsf{S}, \hat{+})$ is a Kegelspitze S equipped with an operation $\hat{+} : \mathbb{R}^{+\infty} \times \mathsf{S} \to \mathsf{S}$ that is ω-continuous in both arguments and satisfies the identities

$$0 \mathbin{\hat{+}} s = s \tag{1}$$

$$c \mathbin{\hat{+}} (d \mathbin{\hat{+}} s) = (c + d) \mathbin{\hat{+}} s \tag{2}$$

$$(c_1 \mathbin{\hat{+}} s_1) +_r (c_2 \mathbin{\hat{+}} s_2) = (c_1 +_r c_2) \mathbin{\hat{+}} (s_1 +_r s_2) \tag{3}$$

Example 2.5. For any Kegelspitze S, we get a cost structure $(\mathsf{S}, +_{\mathsf{f}})$ with *forgetful cost addition* defined by $c +_{\mathsf{f}} r \triangleq r$. A more representative example is given by the cost structure $(\mathbb{R}^{+\infty}, +)$, where $+$ is the standard addition in $\mathbb{R}^{+\infty}$.

3 Quantum Programming Language

We introduce the syntax and operational semantics of our imperative programming language supporting both quantum and classical programming primitives. The syntax is summarised in Figure 1. The `consume(a)` expression is used to represent resource consumption and may be thought of as a meta-language primitive. We model the dynamics (operational semantics, omitted here) of our language as a probabilistic abstract reduction system [3] — a transition system where reducts are chosen from a probability distribution. Reductions can then be defined as stochastic processes [3], or equivalently, as reduction relations over distributions [1]. We follow the latter approach, unlike the former it permits us to define a notion of expected cost concisely, without much technical overhead [2]. By using the operational semantics, we can precisely formulate the expected cost of a quantum program in purely operational terms, as we do in the full paper.

However, this formulation, like most other operational notions, is not compositional in terms of the syntactic structure of the programs. This motivates the development of our quantum expectation transformer (in the next section) which also allows us to recover the expected cost of a program and which has the added benefit of compositionality.

4 Quantum Expectation Transformers

Expectations are functions from the set of (classical and quantum) memory states to cost structures, i.e., functions in $\mathsf{S}^{\mathsf{State}}$, for a given cost structure S. The *quantum expectation transformer* $\mathtt{qet}[\cdot]\{\cdot\}$ is then defined in terms of a program semantics mapping expectations to expectations in a continuation passing style. Specializing the cost structure yields several quantum expectation transformers such as the *quantum expected value transformer* $\mathtt{qev}_{\mathsf{S}}[\cdot]\{\cdot\}$ and the *quantum expected cost transformer* $\mathtt{qect}[\cdot]\{\cdot\}$. After exhibiting several laws and properties of these transformers, we show their soundness and their adequacy.

Definition 4.1. Let $(\mathsf{S}, \hat{+})$ be a cost structure. The quantum expectation transformer

$$\mathtt{qet}[\cdot]\{\cdot\} : \mathtt{Program} \to \mathsf{S}^{\mathsf{State}} \to \mathsf{S}^{\mathsf{State}}$$

is defined inductively in Figure 2 (we have omitted many explanations for notations here for brevity).

Definition 4.2 (Quantum expectation transformers instances)**.**

1. Taking the cost structure $([0,1], +_{\mathsf{f}})$ yields a weakest precondition transformer

   $$\mathtt{qwp}[\cdot]\{\cdot\} : \mathtt{Program} \to [0,1]^{\mathsf{State}} \to [0,1]^{\mathsf{State}},$$

   for probabilistic pre-condition reasoning.

AExp	$\mathsf{a}, \mathsf{a}_1, \mathsf{a}_2$	::=	$\mathsf{x}^{\mathcal{V}} \mid n \mid \mathsf{a}_1 + \mathsf{a}_2 \mid \mathsf{a}_1 - \mathsf{a}_2 \mid \mathsf{a}_1 \times \mathsf{a}_2$
BExp	$\mathsf{b}, \mathsf{b}_1, \mathsf{b}_2$	::=	$\mathsf{x}^{\mathcal{B}} \mid \mathtt{true} \mid \mathtt{false} \mid \mathsf{a}_1 = \mathsf{a}_2 \mid \mathsf{a}_1 \leq \mathsf{a}_2 \mid \mathsf{a}_1 < \mathsf{a}_2 \mid \neg \mathsf{b} \mid \mathsf{b}_1 \wedge \mathsf{b}_2 \mid \mathsf{b}_1 \vee \mathsf{b}_2$
Exp	$\mathsf{e}, \mathsf{e}_1, \mathsf{e}_2$	::=	$\mathsf{a} \mid \mathsf{b}$
Statement	$\mathsf{stm}, \mathsf{stm}_1, \mathsf{stm}_2$	::=	$\mathtt{skip} \mid \mathsf{x}^{\mathcal{K}} = \mathsf{e}^{\mathcal{K}} \mid \overline{\mathsf{q}} \mathrel{*}= \mathsf{U} \mid \mathsf{x}^{\mathcal{B}} = \mathtt{meas}(\mathsf{q}) \mid \mathtt{consume}(\mathsf{a})$
			$\mid \mathsf{stm}_1; \mathsf{stm}_2 \mid \mathtt{if}(\mathsf{b})\{\mathsf{stm}_1\}\ \mathtt{else}\ \{\mathsf{stm}_2\} \mid \mathtt{while}(\mathsf{b})\{\mathsf{stm}\}$

Figure 1. Syntax of quantum programs.

$$\begin{aligned}
\mathsf{qet}\big[\epsilon\big]\big\{f\big\} &\triangleq f &
\mathsf{qet}\big[\mathsf{x} = \mathtt{meas}(\mathsf{q})\big]\big\{f\big\} &\triangleq f[\mathsf{x} := 0; \mathsf{M}_0^{\mathsf{q}}] +_{p_0^{\mathsf{q}}} f[\mathsf{x} := 1; \mathsf{M}_1^{\mathsf{q}}] \\
\mathsf{qet}\big[\mathtt{skip}\big]\big\{f\big\} &\triangleq f &
\mathsf{qet}\big[\mathtt{consume}(\mathsf{a})\big]\big\{f\big\} &\triangleq \max(\llbracket \mathsf{a} \rrbracket, \underline{0}) \mathbin{\hat{+}} f \\
\mathsf{qet}\big[\mathsf{x} = \mathsf{e}\big]\big\{f\big\} &\triangleq f[\mathsf{x} := \mathsf{e}] &
\mathsf{qet}\big[\mathsf{stm}_1; \mathsf{stm}_2\big]\big\{f\big\} &\triangleq \mathsf{qet}\big[\mathsf{stm}_1\big]\big\{\mathsf{qet}\big[\mathsf{stm}_2\big]\big\{f\big\}\big\} \\
\mathsf{qet}\big[\overline{\mathsf{q}} \mathrel{*}= \mathsf{U}\big]\big\{f\big\} &\triangleq f[\mathsf{U}_{\overline{\mathsf{q}}}] &
\mathsf{qet}\big[\mathtt{if}(\mathsf{b})\{\mathsf{stm}_1\}\ \mathtt{else}\ \{\mathsf{stm}_2\}\big]\big\{f\big\} &\triangleq \mathsf{qet}\big[\mathsf{stm}_1\big]\big\{f\big\} +_{\llbracket \mathsf{b} \rrbracket} \mathsf{qet}\big[\mathsf{stm}_2\big]\big\{f\big\} \\
& & \mathsf{qet}\big[\mathtt{while}(\mathsf{b})\{\mathsf{stm}\}\big]\big\{f\big\} &\triangleq \mathsf{lfp}\left(\lambda F.\mathsf{qet}\big[\mathsf{stm}\big]\big\{F\big\} +_{\llbracket \mathsf{b} \rrbracket} f\right)
\end{aligned}$$

Figure 2. Quantum Expectation Transformer $\mathsf{qet}\big[\cdot\big]\big\{\cdot\big\} : \mathsf{Program} \to \mathsf{S}^{\mathsf{State}} \to \mathsf{S}^{\mathsf{State}}$.

2. Taking the cost structure $(\mathsf{S}, +_{\mathsf{f}})$, for any Kegelspitze S, yields an expected value transformer
$$\mathsf{qev}_{\mathsf{S}}\big[\cdot\big]\big\{\cdot\big\} : \mathsf{Program} \to \mathsf{S}^{\mathsf{State}} \to \mathsf{S}^{\mathsf{State}}.$$
3. Taking the cost structure $(\mathbb{R}^{+\infty}, +)$ yields an expected cost transformer
$$\mathsf{qect}\big[\cdot\big]\big\{\cdot\big\} : \mathsf{Program} \to (\mathbb{R}^{+\infty})^{\mathsf{State}} \to (\mathbb{R}^{+\infty})^{\mathsf{State}}.$$

Next, we give a counterpart to the quantum expectation transformer that is defined via the operational semantics (details omitted here). This is again a function
$$\mathsf{QET}[\cdot]\{\cdot\} : \mathsf{Program} \to \mathsf{S}^{\mathsf{State}} \to \mathsf{S}^{\mathsf{State}},$$
but now its definition is purely operational and it is *not* compositional. The soundness statement follows which states that there is a perfect correspondence between the operational notion just introduced and the one we formulate through our quantum expectation transformers (which is compositional).

Theorem 4.3 (Soundness)**.** *For all* $\mathsf{stm} \in \mathsf{Statement}$, $\sigma \in \mathsf{State}$ *and* $f \in \mathsf{S}^{\mathsf{State}}$, $\mathsf{qet}\big[\mathsf{stm}\big]\big\{f\big\}(\sigma) = \mathsf{QET}[\mathsf{stm}]\{f\}(\sigma)$.

This theorem allows us to easily recover the expected cost and expected value of programs.

Corollary 4.4 (Adequacy)**.** *The following identities hold, for all* $\mathsf{stm} \in \mathsf{Statement}$, $\sigma \in \mathsf{State}$ *and* $f \in \mathsf{S}^{\mathsf{State}}$.

1. $\mathsf{qect}\big[\mathsf{stm}\big]\big\{\underline{0}\big\}(\sigma) = \mathsf{ecost}_{\mathsf{stm}}(\sigma)$*; and*
2. $\mathsf{qev}_{\mathsf{S}}\big[\mathsf{stm}\big]\big\{f\big\}(\sigma) = \mathsf{evalue}_{\mathsf{stm}}(f)(\sigma)$.

4.1 Relationship to Denotational Semantics

As a special case of our quantum expectation transformer, we can define a quantum denotational semantics for our language. A cost structure $\mathcal{K}$ can be constructed, such that our quantum expectation transformer qet from Definition 4.1 yields a *quantum denotational semantics* transformer
$$\mathsf{qev}_{\mathsf{K}}\big[\cdot\big]\big\{\cdot\big\} : \mathsf{Program} \to \mathsf{K}^{\mathsf{State}} \to \mathsf{K}^{\mathsf{State}}.$$

Recall that a quantum denotational semantics consists in giving a mathematical interpretation of program configurations which is invariant under the operational semantics (in a probabilistic sense). This can be obtained from $\mathsf{qev}_{\mathsf{K}}$ by making a suitable choice for the continuation, which we call h (details omitted). Then, a quantum denotational semantics
$$(\!|-|\!) : \mathsf{Conf} \cup \mathsf{State} \to \mathsf{K}$$
can be defined by $(\!|(\mathsf{stm}, \sigma)|\!) \triangleq \mathsf{qev}_{\mathsf{K}}\big[\mathsf{stm}\big]\big\{h\big\}(\sigma)$ for configurations and $(\!|\sigma|\!) \triangleq h(\sigma)$ for program states (which are our notion of terminal objects). Then, by Corollary 4.4, for any well-formed configuration $\mu = (\mathsf{stm}, \sigma)$ we have that
$$(\!|\mu|\!) = \mathsf{qev}_{\mathsf{K}}\big[\mathsf{stm}\big]\big\{h\big\}(\sigma) = \mathsf{evalue}_{\mathsf{stm}}(h)(\sigma) = \mathbb{E}_{\mathsf{nf}_{\to}(\mu)}((\!|-|\!)).$$
This shows that the denotational interpretation $(\!|\mu|\!)$ is equal to the (countable) convex sum of the interpretations of final states (i.e., terminal objects) that μ can reduce to, where each probability weight associated to a final state τ is given by the reduction probability of μ to τ as determined by the operational semantics. This is precisely the statement of *strong adequacy* in the denotational semantics of probabilistic [6, 8] and quantum programming languages [5, 9].

References

[1] Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. 2020. On probabilistic term rewriting. *Sci. Comput. Program.* 185 (2020). https://doi.org/10.1016/j.scico.2019.102338

[2] Martin Avanzini, Georg Moser, and Michael Schaper. 2020. A modular cost analysis for probabilistic programs. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 172:1–172:30. https://doi.org/10.1145/3428240

[3] Olivier Bournez and Florent Garnier. 2005. Proving Positive Almost-Sure Termination. In *Proc. of 16th RTA (LNCS, Vol. 3467)*. Springer, 323–337. https://doi.org/10.1142/S0129054112400588

[4] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* 103 (Oct 2009), 150502. Issue 15. https://doi.org/10.1103/PhysRevLett.103.150502

[5] Xiaodong Jia, Andre Kornell, Bert Lindenhovius, Michael Mislove, and Vladimir Zamdzhiev. 2022. Semantics for Variational Quantum Programming. *Proc. ACM Program. Lang.* 6, POPL, Article 26 (jan 2022), 31 pages. https://doi.org/10.1145/3498687

[6] Xiaodong Jia, Bert Lindenhovius, Michael W. Mislove, and Vladimir Zamdzhiev. 2021. Commutative Monads for Probabilistic Programming Languages. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 1–14. https://doi.org/10.1109/LICS52264.2021.9470611

[7] Klaus Keimel and Gordon D. Plotkin. 2017. Mixed powerdomains for probability and nondeterminism. *Log. Methods Comput. Sci.* 13, 1 (2017). https://doi.org/10.23638/LMCS-13(1:2)2017

[8] Thomas Leventis and Michele Pagani. 2019. Strong Adequacy and Untyped Full-Abstraction for Probabilistic Coherence Spaces. In *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11425)*, Mikolaj Bojanczyk and Alex Simpson (Eds.). Springer, 365–381. https://doi.org/10.1007/978-3-030-17127-8_21

[9] Romain Péchoux, Simon Perdrix, Mathys Rennela, and Vladimir Zamdzhiev. 2020. Quantum Programming with Inductive Datatypes: Causality and Affine Type Theory. In *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12077)*, Jean Goubault-Larrecq and Barbara König (Eds.). Springer, 562–581. https://doi.org/10.1007/978-3-030-45231-5_29

[10] Mathys Rennela. 2020. Convexity and Order in Probabilistic Call-by-Name FPC. *Log. Methods Comput. Sci.* 16, 4 (2020). https://lmcs.episciences.org/6901

[11] Peter Selinger. 2004. Towards a quantum programming language. *Math. Struct. Comput. Sci.* 14, 4 (2004), 527–586. https://doi.org/10.1017/S0960129504004256

[12] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332. https://doi.org/10.1137/S0036144598347011