Towards Quantum Control

From superposition of programs to quantum recursion

Daniel Carvalho

INESC TEC/ Universidade do Minho

April 12, 2019

QDays 2019
QuantaLab Workshop in Quantum Computation

Motivation

In a Quantum Turing Machine the flow of execution is described by a constant unitary operator.

Developing a notion of quantum control which permits the superposition of finitely many quantum operations.

Building a programming construct which simplifies the presentation of several quantum algorithms, preserving intuition.

Classical Alternaton

- We are all used to the ever present "if ... then ... else ..." statement, which extracts a boolean value from a predicate and, depending on its truth value, executes one of two statements.

$$\textbf{if } \mathsf{P(b)} \textbf{ then } \mathsf{T} \textbf{ else } \mathsf{Q}$$

- Here b is a bit.

Classical Alternation of Quantum Programs

- The next statement for execution depends on a measurement outcome.
- Construct of probabilistic nature.
- Still resorts to classical information. Therefore not quantum.

$$\begin{aligned} \textbf{measure } M[q] = \; & m_1 \rightarrow P_1 \\ & m2 \rightarrow P_2 \\ & \ldots \\ & m_n \rightarrow P_n \\ \textbf{end} \; & \end{aligned}$$

- Here q is a family of qubits.

What if ...

- Recall the definition of classical alternation but with the following changes:

$$
\begin{aligned}
&q \leftarrow |0\rangle \\
&q \leftarrow H[q] \\
&\textbf{qif } q \textbf{ then } P \textbf{ else } Q
\end{aligned}
$$

- What can we say about **qif** and the behaviour of this program?

Quantum Alternation

Closed Quantum Systems

- Let H be a Hilbert space and let $U_0, U_1 : H \to: H$ be unitary operators. Given a qubit $\mathbf{q}$ define the alternation $Alt_q(U_0, U_1)$ with respect to $\mathbf{q}$ by

$$Alt_q(U_0, U_1) = \Pi_0 \otimes U_0 + \Pi_1 \otimes U_1$$

- It can be represented by a diagonal matrix

$$Alt_q(U_0, U_1) = |0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes U_1 = \begin{pmatrix} U_0 & 0 \\ 0 & U_1 \end{pmatrix}$$

- Intuitively, quantum alternation creates a superposition of execution paths of U_0, U_1 controlled by the basis states of $\mathbf{q}$.

Examples

- Controlled unitary operations can be written in terms of quantum alternation

$$\textbf{qif } q_0 \textbf{ then skip else } q_1 \leftarrow U$$

- One can also write the toffoli gate as

$$\textbf{qif } q_0 \textbf{ then skip else qif } q_1 \textbf{ then skip else } q_2 \leftarrow N$$

- as well as the quantum fourier transform

$$\begin{array}{l} \textbf{for } i = 1\textbf{to } n \textbf{ do} \\ \quad q_i \leftarrow H \\ \quad \textbf{for } k = 2\textbf{to } n - i + 1 \textbf{ do} \\ \quad\quad \textbf{qif } q_{k+i-1} \textbf{ then skip else } q_i \leftarrow R_k \end{array}$$

Here R_k is the phase shift gate defined by $R_k = \Pi_0 + e^{i\theta}\Pi_1$ with $\theta = 2\pi/2^k$.

Open Quantum Systems

- States are density operators, and quantum operations are given by superoperators.
- Superoperators represent the most general evolution that an open quantum system can undergo.
- Mathematically superoperators are described by completely positive trace-preserving maps.
- One way to picture superoperators is as the composition of three maps.

$$
\begin{tikzcd}
\rho_S(0) \otimes \rho_B(0) \arrow[r, "\mathcal{U}"] & U\left[\rho_S(0) \otimes \rho_B(0)\right] U^{\dagger} \arrow[d, "\mathrm{Tr}_B"] \\
\rho_S(0) \arrow[u, "\mathcal{A}"] \arrow[r, dashed, "\Phi"] & \rho_S(t)
\end{tikzcd}
$$

Figure: Superoperator

Kraus representation theorem

• Any superoperator can be written in the form

$$T(\rho) = \sum_k E_k \rho E_k^\dagger$$

• $E_k = \langle e_k | U | e_0 \rangle$ is an operator on the state space of the state space of the principal system.

• The operators $\{E_k\}$ are known as Kraus decompositions of the quantum operation T. These operators must satisfie the completeness relation

$$\sum_k E_k^\dagger E_k \leq I$$

• These decompositions are not unique. One says that two Kraus decompositions are extensionally equal if they represent the same superoperator.

Löwner order on density matrices

• Let D_n be the set of density matrices of dimension n:
$D_\sigma = \{A \in \mathbb{C}^{n\times n} | \ A \ positive \ hermitian \ and \ tr \ A \leq 1\}$

• For matrices $A, B \in \mathbb{C}^{n\times n}$, define $A \sqsubseteq B$ if the matrix $B - A$ is positive. It follows that $\sqsubseteq$ defines a partial order.

• Moreover the poset $(D_n, \sqsubseteq)$ is a complete partial order, i.e, it has least upper bounds of increasing sequences.

• Least upper bounds of increasing sequences coincide with topological limits in the Euclidean topology.

• Any order preserving function on operators will preserve lubs of increasing sequences if it is topologically continuous.

QPL Programming language

Syntax

• high-level features such as loops, recursive procedures, and structured data types.

• functional in nature, statically typed with denotational semantics in terms of a complete partial order of superoperators.

$$
\begin{aligned}
QPL\ Terms\ P, Q ::= \ & \textbf{new bit}\ b := 0 \mid \textbf{new qbit}\ q := 0 \mid \textbf{discard}\ x \\
& \mid b := 0 \mid b := 1 \mid q_1, ..., q_n * = S \\
& \mid \textbf{skip} \mid P; Q \\
& \mid \textbf{if}\ b\ \textbf{then}\ P\ \textbf{else}\ Q \mid \textbf{measure}\ q\ then\ P\ else\ Q \\
& \mid \textbf{while}\ b\ \textbf{do}\ P \\
& \mid \textbf{proc}\ X : \Gamma \rightarrow \Gamma'\{P\}\ \textbf{in}\ Q
\end{aligned}
$$

• A state for a typing context Γ containing n bits and m qubits is given by a $2^n - tuple$ $(A_0, ..., A_{2^n-1})$ of density matrices.

QPL Programming Language

Formal Semantics

- We can assign to QPL a category $\mathbf{Q}$ which has as its objects tuples of density matrices and whose morphisms are precisely superoperators.

- Identity morphisms are superoperators and, superoperatores are closed under composition.

- $\mathbf{Q}$ is equipped with two categorical operations, concatenation and tensor product.

- The Löwner order is naturally extended to matrix tuples. This makes $\mathbf{Q}(\sigma, \sigma')$ into a complete partial order.

QPL Programming Language

Q atomic morphisms

- The atomic statements in QPL are interpreted as the following morphisms in **Q**,

$$
\begin{array}{llll}
[\![\text{new bit } b := \mathbf{0}]\!] & = \ \mathit{newbit} : & \mathbf{I} \to \mathbf{bit} : & a \mapsto (a, 0) \\
[\![\text{new qbit } q := \mathbf{0}]\!] & = \ \mathit{newqbit} : & \mathbf{I} \to \mathbf{qbit} : & a \mapsto \begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix} \\
[\![\text{discard } b]\!] & = \ \mathit{discardbit} : & \mathbf{bit} \to \mathbf{I} : & (a, b) \mapsto a + b \\
[\![\text{discard } q]\!] & = \ \mathit{discardqbit} : & \mathbf{qbit} \to \mathbf{I} : & \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto a + d \\
[\![b := \mathbf{0}]\!] & = \ \mathit{set}_0 : & \mathbf{bit} \to \mathbf{bit} : & (a, b) \mapsto (a + b, 0) \\
[\![b := \mathbf{1}]\!] & = \ \mathit{set}_1 : & \mathbf{bit} \to \mathbf{bit} : & (a, b) \mapsto (0, a + b) \\
[\![\bar{q} \mathrel{*}= S]\!] & = \ \mathit{unitary}_S : & \mathbf{qbit}^n \to \mathbf{qbit}^n : & A \mapsto SAS^* \\
[\![\text{branch } b]\!] & = \ \mathit{branch} : & \mathbf{bit} \to \mathbf{bit} \oplus \mathbf{bit} : & (a, b) \mapsto (a, 0, 0, b) \\
[\![\text{measure } q]\!] & = \ \mathit{measure} : & \mathbf{qbit} \to \mathbf{qbit} \oplus \mathbf{qbit} : & \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \left(\begin{pmatrix} a & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & d \end{pmatrix} \right)
\end{array}
$$

Figure: Morphisms in Q

QPL Programming language

Recursion

- The recursion can be partially unwound.
- The successive unwindings are given by $F(0), F^2(0), \ldots$
- Each unwinding is less than the next in the Löwner order, because F is monotone.
- The recursion meaning is given by a least upper bound of the increasing sequence.
- Because density matrices form a complete partial order we are sure that the lubs exist.

Quantum Alternation

• We can define Quantum Alternation in terms of superoperators as follows :

Given a qubit **q** and two superoperators $T_0, T_1 : S(H) \to S(K)$ then the alternation of T_0 and T_1 with respect to **q** should be the superoperator $Alt_{\mathbf{q}}(T_0, T_1) : S(\mathbf{qbit} \otimes H) \to S(\mathbf{qbit} \otimes K)$,

$$Alt_q(T_0, T_1) :: \rho = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \mapsto \begin{bmatrix} T_0 A & * \\ * & T_1 D \end{bmatrix}$$

ρ is a state on $\mathbf{qbit} \otimes H$.

• Quantum alternation should use the information incoded in the classical states of q. If the qubit **q** is in a classical state Π_i with i $\in \{0, 1\}$, the $\{Alt_q(T_0, T_1)\} = I \otimes T_i$. The alternation reduces to a local operation on $\mathbf{qbit} \otimes S$(H).

• One general way of defining the semantics of quantum alternation would be in terms of Kraus decompositions.

• Define a new category **K** with Hilbert spaces as objects and Kraus decompositions as morphisms.

• Composition $S\ o\ T$ is defined to be the set obtained from the multiset $X = \{E\ .\ F | E \in S, F \in T\}$ by replacing l ocurrences of operator $K \in X$ with $\sqrt{l}K$.

• Identity is the singleton set containing the usual Identity operator $id_H = \{I\}$.

Quantum Alternation

Kraus Semantics

- Finally we are in a position to define quantum alternation of two morphisms $S, T : H \to K$ to be the morphism $Alt_q(S, T) : qbit \otimes H \to qbit \otimes K$:

$$Alt_q(S, T) = \left\{ \Pi_0 \otimes \frac{E}{\sqrt{|T|}} + \Pi_1 \otimes \frac{F}{\sqrt{|S|}} \middle| E \in S, F \in T \right\}$$

QPL and Quantum Alternation

- The semantics of quantum alternation are given by Kraus decompositions.
- The sematics of quantum alternation cannot be lifted to semantics of superoperators because Quantum alternation does not preserve extensional equality.
- Quantum alternation is not compatible with recursion defined in QPL because quantum alternation is not monotone with respect to the Löwner order.

A different approach

Random walks

The simplest random walk is the one-dimensional walk in which a particle moves ona lattice marked by integers $\mathbb{Z}$, and at each step it moves one position left or right, depending on the flip of a fair coin.

Quantum Walks

- Hilbert space $H_d \otimes H_p$.
- $H_d = span\{|L\rangle, |D\rangle\}$ denotes the direction space.
- $H_p = span\{|n\rangle : n \in \mathbb{Z}\}$ denotes the position space.
- One step of the walk is represented by the unitary operator $W = T(H \otimes I)$.
- T is a unitary operator in $H_d \otimes H_p$ defined by

$$T|L, n\rangle = |L, n-1\rangle \quad T|R, n\rangle = |R, n+1\rangle$$

- H is the Hadamard transform in the direction space H_d,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- Quantum alternation is disguised in the quantum walk.

- Consider the operators in position space H_p

$$T_L|n\rangle = |n-1\rangle \quad T_R|n\rangle = |n+1\rangle$$

- The translation operator T can be written as

$$\textbf{qif } d \textbf{ then } T_L[p] \textbf{ else } T_R[p]$$

- d represents the direction space and is employed by an external coin system. p represents the position space which we denote as the principal system.

- Furthermore, the single-step walk operator W can be seen as

$$\begin{array}{l} d \leftarrow H[d] \\ \textbf{qif } d \textbf{ then } T_L[p] \textbf{ else } T_R[p] \end{array}$$

Quantum Recursion

- We already established that quantum alternation is not compatible with recursion defined in QPL.

- QPL recursion is defined with procedure calls controlled classicaly.

- We may consider this type of recursion as classical recursion of quantum programs.

- Is there a quantum counterpart?

- Consider the program

$$X \Leftarrow H[d]; \textbf{qif } d \textbf{ then } (T_L[p]; X) \textbf{ else } (T_R[p]; X)$$

Quantum Recursion

- This program represents a recursive quantum walk.

- The major difference between random and quantum walks is caused by quantum interference.

- Recursive quantum walks exibit a higher-level interference, i.e., interference between infinite paths. Paths containing the recursive walk itself may cancell each other.

- How do we solve the recursive quantum equation above?

Quantum Recursion

• The number of coin "particles" needed in running a recursive walk is unknown beforehand.

• Need for a method that describes quantum systems with variable number of identical particles.

• Solution: Second Quantization!

• Principle of symmetrisation: States of n identical particles are either completely symmetric or completely antisymmetric with respect to the permutation of the particles.

• Implementation will depend on the specific choice of coin "particle" : bosons or fermions.

Examples

Quantum while loops

- In classical programming the while-loop

$$\textbf{while } \mathrm{b} \textbf{ do } \mathrm{S}$$

can be written as the recursive program

$$X \Leftarrow \textbf{if } b \textbf{ then } X \textbf{ else } \mathit{skip}$$

- Similarly one can write a quantum version

$$\textbf{qwhile } [\mathrm{c}] = |1\rangle \textbf{ do } U[\mathrm{q}]$$

as

$$X \Leftarrow \textbf{if } c \textbf{ then } \mathit{skip} \textbf{ else } U[q]; X$$

- Obviously this two versions are equivalent.

Examples

Quantum while loops

- A more intersting quantum loop would be

$$\textbf{qwhile } \mathrm{V[c]} = |1\rangle \textbf{ do } U\mathrm{[q]}$$

as well as

$$\textbf{qwhile } \mathrm{W[c;q]} = |1\rangle \textbf{ do } U\mathrm{[q]}$$

References

[1] Mingsheng Ying (2014) : Quantum Recursion and Second Quantization.

[2] Mingsheng Ying, Nengkun Yu & Yuan Feng (2014) : Alternation on quantum programming: from superposition of data to superposition of programs.

[3] Prakash Panangaden & Costin Bădescu (2015) : Quantum Alternation: Prospects and Problems.

[4] Peter Selinger (2004) : Towards a Quantum Programming Language.

[5] K. Kraus (1983): States, Effects and Operations. Lecture Notes in Physics 190, Springer-Verlag.