The Quantum IO Monad

QIO

Alexander S. Green
asg@cs.nott.ac.uk

Foundations of Programming Group,
School of Computer Science & IT,
University of Nottingham

Quantum Computation

- Quantum computation uses Qubits .

Quantum Computation

- Quantum computation uses Qubits .
- Qubits have 2 base states ($|0\rangle$ and $|1\rangle$), but can exist in a *superposition* of both states simultaneously, until measured.

Quantum Computation

- Quantum computation uses Qubits .
- Qubits have 2 base states ($|0\rangle$ and $|1\rangle$), but can exist in a *superposition* of both states simultaneously, until measured.
- Multiple qubits can become entangled, meaning that an n-qubit system has 2^n base states, and can be in a superposition of all these 2^n states.

Quantum Computation

The University of Nottingham

- For example, a 2 qubit system has the base states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.

Quantum Computation

- For example, a 2 qubit system has the base states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.
- An arbitrary state ($|\phi\rangle$) can be described by $|\phi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$ where $\alpha, \beta, \gamma, \delta \in \mathbf{C}$ are the complex amplitutes of each base state.

Quantum Computation

- For example, a 2 qubit system has the base states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.
- An arbitrary state ($|\phi\rangle$) can be described by $|\phi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$ where $\alpha, \beta, \gamma, \delta \in \mathbf{C}$ are the complex amplitutes of each base state.
- When measured the state collapses into one of the base states. Where the probability of it collapsing into each state is $|\alpha|^2$, $|\beta|^2$, $|\gamma|^2$ or $|\delta|^2$ respectively.

Quantum Computation

The University of Nottingham

- For example, a 2 qubit system has the base states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.
- An arbitrary state ($|\phi\rangle$) can be described by $|\phi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$ where $\alpha, \beta, \gamma, \delta \in \mathbf{C}$ are the complex amplitutes of each base state.
- When measured the state collapses into one of the base states. Where the probability of it collapsing into each state is $|\alpha|^2$, $|\beta|^2$, $|\gamma|^2$ or $|\delta|^2$ respectively.
- Thus the condition that $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$ always holds.

Quantum Computation

- Quantum algorithms are designed to take advantage of these entangled states.

Quantum Computation

- Quantum algorithms are designed to take advantage of these entangled states.
- Shor's algorithm, is a quantum algorithm for factoring large numbers. It has an exponential speed up over the fastest known classical algorithm.

Quantum Computation

- Quantum algorithms are designed to take advantage of these entangled states.
- Shor's algorithm, is a quantum algorithm for factoring large numbers. It has an exponential speed up over the fastest known classical algorithm.
- Grover's algorithm, is a quantum algorithm for searching an unsorted database. It has quadratic speed up over the linear search, which is provably the fastest classical algorithm.

Quantum Computation

- Quantum algorithms are designed to take advantage of these entangled states.
- Shor's algorithm, is a quantum algorithm for factoring large numbers. It has an exponential speed up over the fastest known classical algorithm.
- Grover's algorithm, is a quantum algorithm for searching an unsorted database. It has quadratic speed up over the linear search, which is provably the fastest classical algorithm.
- Arbitrary qubit states cannot be copied, however they may teleported to another qubit using entangled pairs known as Bell states.

Haskell

The University of Nottingham

- We'd like to introduce Quantum Computation to Functional Programmers.

Haskell

The University of Nottingham

- We'd like to introduce Quantum Computation to Functional Programmers.
- and Functional Programming to the Quantum community!

Haskell

- We'd like to introduce Quantum Computation to Functional Programmers.
- and Functional Programming to the Quantum community!
- Haskell is a Pure functional programming language, and thus introduces Monads to encapsulate effects.

Haskell

The University of Nottingham

- We'd like to introduce Quantum Computation to Functional Programmers.
- and Functional Programming to the Quantum community!
- Haskell is a Pure functional programming language, and thus introduces Monads to encapsulate effects.
- For example, the IO Monad.

Monads

$$\mathbf{class}\ Monad\ m\ \mathbf{where}$$
$$(\gg\!=) \in m\ a \rightarrow (a \rightarrow m\ b) \rightarrow m\ b$$
$$return \in a \rightarrow m\ a$$

such that the following equations hold

$$return\ a \gg\!= f = f\ a$$
$$c \gg\!= return = c$$
$$(c \gg\!= f) \gg\!= g = c \gg\!= \lambda a \rightarrow f\ a \gg\!= g$$

The IO Monad

- All IO in Haskell takes place within the IO Monad.

The IO Monad

- All IO in Haskell takes place within the IO Monad.
- For example, echoing a Character to the Screen

$$getChar \in IO\ Char$$

$$putChar \in Char \rightarrow IO\ ()$$

$$echo \in IO\ ()$$

$$echo = getChar \gg= (\lambda c \rightarrow putChar\ c) >> echo$$

The IO Monad

- All IO in Haskell takes place within the IO Monad.
- For example, echoing a Character to the Screen

$$getChar \in IO\ Char$$

$$putChar \in Char \rightarrow IO\ ()$$

$$echo \in IO\ ()$$

$$echo = getChar \ggg= (\lambda c \rightarrow putChar\ c) >> echo$$

- Haskell provides sytactic sugar for Monadic Programming. (in the form of do notation)

$$\begin{aligned} echo = \mathbf{do}\ & c \leftarrow getChar \\ & putChar\ c \\ & echo \end{aligned}$$

The QIO Monad

The University of Nottingham

- We'd like to think of a quantum computer as a classical computer, along with a register of qubits.

The QIO Monad

- We'd like to think of a quantum computer as a classical computer, along with a register of qubits.
- The quantum register would execute any quantum parts of a computation.

The QIO Monad

- We'd like to think of a quantum computer as a classical computer, along with a register of qubits.
- The quantum register would execute any quantum parts of a computation.
- However, quantum registers don't exist (yet?)!!!

The QIO Monad

- We'd like to think of a quantum computer as a classical computer, along with a register of qubits.
- The quantum register would execute any quantum parts of a computation.
- However, quantum registers don't exist (yet?)!!!
- So the QIO Monad can be used to encapsulate the behaviour that would be given by a quantum register.

The QIO Monad

The University of Nottingham

- We can use the QIO Monad to construct Quantum Computations

The QIO Monad

- We can use the QIO Monad to construct Quantum Computations
- The QIO Monad has two ways of evaluating a Quantum Computation.

The QIO Monad

- We can use the QIO Monad to construct Quantum Computations
- The QIO Monad has two ways of evaluating a Quantum Computation.
- The first uses a random number generator to measure the qubits, so the outcome is equivalent to running the quantum computation.

The QIO Monad

- We can use the QIO Monad to construct Quantum Computations
- The QIO Monad has two ways of evaluating a Quantum Computation.
- The first uses a random number generator to measure the qubits, so the outcome is equivalent to running the quantum computation.
- The second gives the state of the quantum register after evaluating the program, e.g. a list of probabilities of the base states that could be reached on measuring.

The QIO Monad

The University of Nottingham

- We can use the QIO Monad to construct Quantum Computations
- The QIO Monad has two ways of evaluating a Quantum Computation.
- The first uses a random number generator to measure the qubits, so the outcome is equivalent to running the quantum computation.
- The second gives the state of the quantum register after evaluating the program, e.g. a list of probabilities of the base states that could be reached on measuring.
- A quantum computation can be constructed in the QIO Monad (using do notation), and then evaluated using either of the available evaluators.

QIO Computations

$$
\begin{aligned}
&rbit \in QIO\ Bool \\
&rbit = \mathbf{do}\ x \leftarrow mkQbit \\
&\qquad\qquad applyU\ (rotate\ x\ rh) \\
&\qquad\qquad b \leftarrow meas\ x \\
&\qquad\qquad return\ b
\end{aligned}
$$

QIO Computations

$$
\begin{aligned}
&bell \in QIO\ (Bool, Bool)\\
&bell = \mathbf{do}\ x \leftarrow mkQbit\\
&\qquad\quad applyU\ (rotate\ x\ rh)\\
&\qquad\quad y \leftarrow mkQbit\\
&\qquad\quad applyU\ (x | rotate\ y\ rx)\\
&\qquad\quad b \leftarrow meas\ x\\
&\qquad\quad c \leftarrow meas\ y\\
&\qquad\quad return\ (b, c)
\end{aligned}
$$

The future of the QIO monad

The University of Nottingham

- Now that we have created the QIO Monad, we would like to come up with larger examples, including implementations of Shor's and Grover's algorithms.

The future of the QIO monad

- Now that we have created the QIO Monad, we would like to come up with larger examples, including implementations of Shor's and Grover's algorithms.
- It should also be possible to use larger quantum data structures than individual qubits, creating them in the same way that classical data structures are defined from classical bits.

The future of the QIO monad

The University of Nottingham

- Now that we have created the QIO Monad, we would like to come up with larger examples, including implementations of Shor's and Grover's algorithms.
- It should also be possible to use larger quantum data structures than individual qubits, creating them in the same way that classical data structures are defined from classical bits.
- It should also be possible to construct QIO programs from QML programs.

The End

Thank you all for listening!