

Quantum programming languages

Jaroslav Mischczak

Institute of Theoretical and Applied Informatics,
Polish Academy of Sciences

26 September 2007

- 1 Introduction
- 2 QCL basics
- 3 Quantum data types
 - State initialization
 - Radix sort
- 4 Final remarks

- All the data that we process on a computer ultimately decompose into individual bits, but writing programs that exclusively process bits would be tiresome indeed.¹

¹R. Sedgewick, *Algorithms in Java*, Addison Wesley (2002)

²P. W. Shor, *Progress in Quantum Algorithms*, Quantum Information Processing, **3**. (2004)

- All the data that we process on a computer ultimately decompose into individual bits, but writing programs that exclusively process bits would be tiresome indeed.¹
- The second [reason that quantum algorithms might be difficult to discover] is that quantum computers operate in a manner so non-intuitive, and so different from classical computers, that all the experience of the last fifty years in discovering classical algorithms offers little insight into how to go about finding quantum algorithms [...]²

¹R. Sedgewick, *Algorithms in Java*, Addison Wesley (2002)

²P. W. Shor, *Progress in Qantum Algorithms*, Quantum Information Processing, **3**. (2004)

QCL – Quantum Computation Language

- Provides quantum registers for data manipulation

```
qcl> qureg a[2];  
qcl> H(a[1]);  
[2/64] 0.70711 |0> + 0.70711 |2>
```

- Allows for quantum conditional statements

```
qcl> qureg a[1];  
qcl> H(a);  
[1/64] 0.70711 |0> + 0.70711 |1>  
qcl> if (a) print "OK";  
: OK
```

- Introduces special types of functions for dealing with the management of quantum memory

Limitations of QCL

- Only one data structure – qreg is an array of qubits

Limitations of QCL

- Only one data structure – qreg is an array of qubits
- No means for defining new data types

Limitations of QCL

- Only one data structure – qreg is an array of qubits
- No means for defining new data types
- Does not provide operators for quantum-specific operations

Quantum data types – some examples

- State initialization – create flat superposition of the elements (integers) from some set.
- Radix sorting – create superposition of the elements from the given set such that after measurement it is possible to obtain information about ordering.

Example 1: State initialization (1/2)

Initialize some variable as a superposition of integers – à la Perl6 junctions.

```
qint q1 = (1|3|5|7);
```

Problem

For a given set of integers $A = \{a^1, a^2, \dots, a^K\}$ output unitary operation $R(A)$ such that

$$R(A)|0 \dots 0\rangle = \sum_{i=1}^K |a^i\rangle. \quad (1)$$

Where $\alpha_k(a^i)$ – binary string of the length $M - k$ composed of binary numbers a'_{k+1}, \dots, a'_M .

Example 1: State initialization (2/2)

- 1 Perform rotation $R_1 = G(t_1/K) \otimes \mathbb{I}^{M-1}$

$t_k(\alpha_j)$ – number of occurrences of 1 at the k -th bit of the input numbers for control α_j .

Example 1: State initialization (2/2)

- 1 Perform rotation $R_1 = G(t_1/K) \otimes \mathbb{I}^{M-1}$
- 2 For each qubit from $k = 2$ to N perform rotation

$$R_k = \left(\prod_{\alpha_j} X_k(\alpha_j) \right) \otimes \mathbb{I}^{M-k-1},$$

where

$$X_k(\alpha_j) = |\alpha_j\rangle\langle\alpha_j| \otimes G\left(\frac{t_k(\alpha_j)}{c_k(\alpha_j)}\right) + (\mathbb{I}^k - |\alpha_j\rangle\langle\alpha_j|) \otimes \mathbb{I}.$$



$t_k(\alpha_j)$ – number of occurrences of 1 at the k -th bit of the input numbers for control α_j .

Example 2: Sorting algorithm

Example 2: Sorting algorithm – complexity

After $O(n^2)$ executions of the gate Q we can obtain sufficient information about the probability distribution and restore information about the mutual ordering of the input elements.

Bibliography

-  B. Ömer, *Structured Quantum Programming*. PhD thesis, Technical University of Vienna (2003).
-  M. Mottonen *et al*, *Transformation of quantum states using uniformly controlled rotations*. *Quantum Information & Computation* **5**(6) (2005)