- Why aren’t there more quantum algorithms ?
- Quantum Programming Languages

By : Amanda Cieslak and Ahmana Tarin

Why aren’t there more quantum algorithms?

- there are only a few problems for which quantum computers can offer exponential speed-up over classical computers (Shor’s, Grover’s)
- difficult to go about finding a quantum algorithm compared to classical algorithms because quantum computers are very different than classical computers, so the approach to an algorithm is very different too.

Unique Problems

- Researchers work and look for quantum algorithms that solve problems that are not known to be solved classically in polynomial time, which is difficult since to find such a problem we would assume quantum computers can not solve NP-complete problems in faster exponential time as opposed to classical computers. (Superpolynomial speed-up cannot arise from problems that have polynomial-time classical algorithms, like P AND NP).
- In order to achieve super-polynomial speed-up, we would need to find a problem which is neither in P nor is NP-complete. Due to the success of classical algorithms, there are few natural problems that fit these requirements.

- most of the focus in classical computing has been on classifying problems as polynomial time or as NP-hard, which has left few well-studied problems outside that are not known to be outside of these two classes.
- Also searching for super-polynomial speed-up is difficult, and what might be a more useful approach is to try to find faster quantum algorithms for problems already known to be classically solvable in polynomial time. This provides polynomial factor speed-ups, which can lead to finding other new techniques for designing quantum algorithms.

Applying quantum physics

- computer scientists have less experience with quantum mechanics in comparison to physicists. In order for a quantum algorithm to provide speed-up over a classical computation it must use interference, which is a concept less known by computer scientists.

Quantum Programming Languages

- A programming language for quantum hardware to help us control the quantum computing device and implement a quantum algorithm
- To provide tools for researchers to understand how quantum computation works and how to reason formally about quantum algorithms
- Classical Programming : DATA + CONTROL = PROGRAMMING
- Quantum Programming: QUANTUM DATA + CONTROL = QUANTUM PROGRAMMING
- This gives us the option to manipulate quantum data without having to have the knowledge of the underlying operating system and quantum hardware

Architectures for Quantum Computing

1. Quantum Circuits:
    a. Input Device - where we can feed in quantum data
    b. Basic Gates - quantum circuits
    c. Device to measure - result is a sequence of bits that can be stored/manipulated later
2. Quantum Turing Machines:
    a. A quantum analog of Turing Machines
    b. Convenient for discussing quantum complexity, but not a design for a programming language
3. Quantum Random Access Memory Model (QRAM):
    a. A classical computer, playing the role of the master
    b. A quantum computer device, that can be accessed by the master computer on request

QRAM

- Idea: A programmer writes classical code in a classical language, but when they need the extra quantum power, they can add a few lines of quantum assembler code.
- No need to know how qubits are physically stored, initialized, manipulated, or measured.
- Quantum Hardware Interface, QHI, will translate assembler commands issued by the master into explicit actions performed by the quantum device.

Quantum Processor

1. A set of quantum data storage registers
   a. Quantum Register: An interface to an addressable sequence of qubits. Each q-register as a unique identifier by which it is referred.
   b. The first thing to do is ask the quantum device through the quantum hardware interface to initialize a sequence of qubits
2. Utilities that apply operations on the storage

- After the quantum register has been initialized and manipulated, the programmer can issue a command that will measure selected portions. This will return a classical value to the main program.

Quantum Programming

INITIALIZE R 2

- allocated a 2 qubit register names R and initializes it to |00>

U TENSOR H H

- creates a unitary matrix U of size 4 x 4

APPLY U R

- applies U to R

MEASURE R RES

- measures the q-register R and stores the result in the bit array RES

Languages

- Imperative Programming: program that is mainly a sequence of commands with flow control statements. C, Python, Java
- Logical Programming: a program is a specification of properties and relations in a fragment of first-order logic
- Functional Programming: specifications of a function. The program will be provided with an acceptable value for the function and it will compute the return value. (QUANTUM DATA AND CLASSICAL CONTROL)

- Imperative Quantum Programming Languages:
  - QCL (Quantum Computation Language): high level, architecture independent programming language for quantum computers, with a syntax derived from classical procedural languages like C or Pascal. This allows for the complete implementation and simulation of quantum algorithms (including classical components) in one consistent formalism.
  - Standard Library includes:
    - controlled-not with many target qubits,
    - Hadamard operation on many qubits,
    - parse and controlled phase.
- Functional Quantum Programming Languages
  - QFL and QPL: QFC and QPL are two closely related quantum programming lan
    by Peter Selinger. They differ only in their syntax.
  - These languages have classical control flow but can operate on quantum or classical data.