Printf(“Hello Quantum World!)”; Learn The Essentials Of Quantum Computing

29/01/2019

DISHA MISAL

Found a way to Data Science and AI though her fascination for Technology. Likes to read, watch football and has an enourmous amount affection for Astrophysics.

Intro

- IBM recently unveiled commercial quantum computer called the IBM Q System One. There are already several programming languages dedicated for quantum programming. Here are some examples of quantum programming languages that will help to write quantum algorithms using high-level constructs.

cin>>Quantum Programming – Q#

- Here are some languages for quantum programming.
- **1.Q#:** Q# fro Microsoft is perhaps one of the most popular programming languages among all the other languages in this list. Q# is a production of Microsoft. Through its *Quantum Development Kit,* it has a quantum simulator, libraries to implement quantum algorithms and the language Q#, which is available as a separately downloaded extension for Visual Studio. The language is used for writing subroutine that executes on an adjunct quantum processor, under the control of a classical host program and computer. They currently have the subroutes on Q# executed on a simulator, since quantum processors are not widely available.
- Q# provides a small set of primitive types, along with two ways (arrays and tuples) for creating new, structured types. It supports a basic procedural model for writing programs, with loops and if/then statements. The top-level constructs in Q# are user defined types, operations, and functions. The only Qubit expressions are symbols that are bound to Qubit values or array elements of Qubit arrays. There are no Qubit literals. Following is an example of a Q# code from Microsoft:

```
namespace Quantum.Bell
{
  open Microsoft.Quantum.Canon;
  open Microsoft.Quantum.Primitive;
  operation HelloQ () : Unit {
     Message(“Hello quantum world!”);
  }
}
```

cin>>Quantum Programming – PyQuil

- **2.PyQuil:** A product of Rigetti's software development kit called Forest, which is a developer environment to write and run quantum programs, PyQuil is a Python-based library. It stands for Python Quantum Instruction Language and is an open source Python library hosted on GitHub, to help with writing and running quantum programs. Quantum Instruction Language (Quil) is a quantum instruction language standard. Instructions written in Quil can be executed on any implementation of a quantum abstract machine, such as the quantum virtual machine (QVM), which is cloud-based, or on a real quantum processing unit (QPU). This is a classical simulation of a quantum processor that can simulate various qubit operations. The default access key allows you to run simulations of up to 26 qubits.
- Another two-qubit gate example is the SWAP gate, which swaps the |01⟩ and |10⟩ states:
- SWAP|01> = (1+0j)|10>
- With outcome probabilities
- {'00': 0.0, '01': 0.0, '10': 1.0, '11': 0.0}
- Quantum programs are built by applying successive gate operations:

```
# Composing qubit operations is the same as multiplying matrices sequentially
p = Program(X(0), Y(0), Z(0))
wavefunction = quantum_simulator.wavefunction(p)

print("ZYX|0> = ", wavefunction)
print("With outcome probabilities\n", wavefunction.get_outcome_probs())
ZYX|0> =  [ 0.-1.j 0.+0.j]
With outcome probabilities
{'0': 1.0, '1': 0.0}
```

cin>>Quantum Programming – Open QASM

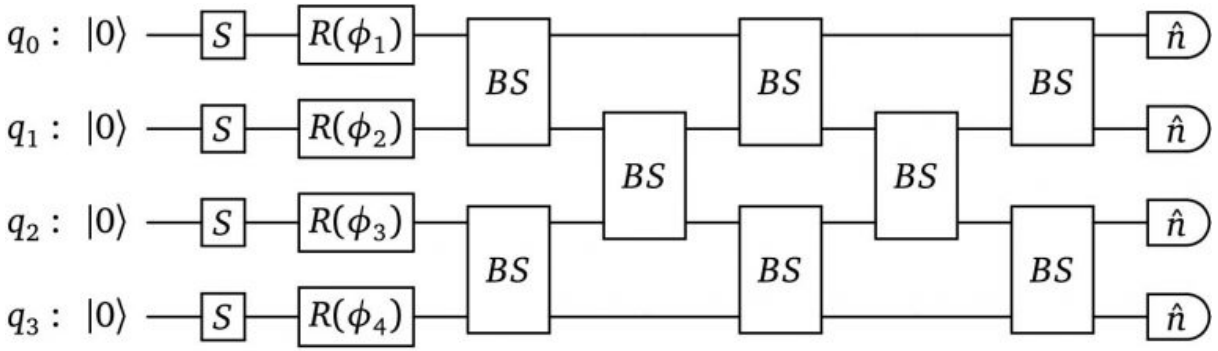
- 3**.OpenQASM:** Open Quantum Assembly Language (OpenQASM) is an intermediate representation for quantum instructions. IBM has a Quantum Information Software Kit (QISKIT) has the source code of OpenQASM and was released in the year 2017. It was released to be used along with IBM's cloud quantum computing platform called IBM Q Experience. The syntax of the human-readable form of Open QASM has elements of C and assembly languages. In this language, Gates are
- defined by statements of the form: (https://arxiv.org/pdf/1707.03429.pdf)

```
gate name(params) qargs
{
body
}
```

cin>>Quantum Programming –BlackBird

- **4.Blackbird:** Xanadu, a full stack quantum startup, in its attempt of quantum computing, has a continuous-variable quantum computation platform called Strawberry Fields. Quantum circuits are written using the easy-to-use and intuitive Blackbird programming language. It is a Python based language. The language can be used to design quantum circuits, design a photonics experiment or do almost everything in the space of quantum programming. They also have their source code on github. They have an interactive web app which allows to run a quantum computing simulation via drag and drop and has quantum computer simulators implemented using NumPy and Tensorflow. Future releases of Strawberry Fields will aim to target experimental backends, including photonic quantum computing chips
- The following is an example of a CV quantum circuit for Gaussian boson sampling:
- Here is an example of the Blackbird quantum circuit language for the above boson sampling circuit, with randomly chosen rotation angles and beamsplitter parameters:

```
# prepare the input squeezed states
S = Sgate(1)
S | q[0]
S | q[1]
S | q[2]
S | q[3]
# linear interferometer
Interferometer(U) | q
```



cin>>Quantum Programming – Quipper, QCL, QFC, QPL

- **5.Quipper:** A high-level circuit description language, it includes gate-by-gate descriptions of circuit fragments, as well as powerful operators for assembling and manipulating circuits. It has a good support for hierarchical circuits. Extensive libraries of quantum functions, such as libraries for quantum integer and fixed-point arithmetic, the Quantum Fourier transform, an efficient Qram implementation, libraries for simulation of pseudo-classical circuits, Stabilizer circuits, and arbitrary circuits, libraries for exact and approximate decomposition of circuits into specific gate sets. This is a Haskell-based language.
- **6.QCL:** Quantum Computing Language (QCL) is one of the most advanced implemented quantum programming language. Its syntax is similar to that of C and classical data types are similar to primitive data types in C. The basic built-in quantum data type in QCL is qreg (quantum register), which can be interpreted as a an array of qubits. A classical code and quantum code can be combined in the same program.
- **7.QFC and QPL:** Introduced by Peter Selinger QFC uses a flowchart syntax, whereas QPL uses a textual syntax, and this forms as their only difference. These languages have classical control flow but can operate on quantum or classical data. Selinger gives a denotational semantics for both languages in a category of superoperators, which is nothing but a linear operator operating on a vector space of linear operators.

END