Quantum Programming Languages: Cirq

Jens Palsberg

May 20, 2019

Outline

Hook: The next round of quantum programming homework is in Cirq. Cirq is a mix of Python and of calls to operations on a quantum machine. The homework is to program and run well-known algorithms.

Purpose: Persuade you that you can get this to work.

Preview:

1. We will cover the Cirq view of qubits, moments, operations, and circuits.
2. We will show how to define new gates in Cirq.
3. We will cover such advanced topics as parameterization, noise, and optimization.

Transition to Body: Let us begin with the basics.

Main Point 1: We will cover the Cirq view of qubits, moments, operations, and circuits.
[Cirq has multiple ways of building circuits]
[Cirq has syntax for running a simulation once or multiple times]
[Cirq gives access to intermediate and final results]

Transition to MP2: Now let us go beyond what is hardwired in Cirq.

Main Point 2: We will show how to define new gates in Cirq.
[We can define a new gate as a class]
[We have to define how it works and how to display it in a diagram]
[We can also specify how to decompose it into a product of gates]

Transition to MP3: Finally let us talk about aspects of Cirq that are different from PyQuil.

Main Point 3: We will cover such advanced topics as parameterization, noise, and optimization.
[Paraterization uses symbolic values that are resolved at run time]
[Cirq supports a noise model that makes a simulator mimic a quantum computer]
[Cirq supports optimization-on-demand]

Transition to Close: So there you have it.

Review: Cirq has a syntax for circuits that is verbose yet also flexible, which is useful for tasks such as definition of new gates. Additionally, Cirq has advanced features that support both programming of large circuits and simulating in a way that mimics running on a quantum computer.

Strong finish: Cirq is developed and supported by a sizable team at Google. Research groups at University of Wisconsin and Duke University are working on making Cirq run on their own quantum computers. Those quantum computers are significantly different from Google's quantum computer, which uses superconducting qubits.

Call to action: Do you like Cirq or PyQuil better?

Detailed presentation

Hook: The next round of quantum programming homework is in Cirq. Cirq is a mix of Python and of calls to operations on a quantum machine. On the surface, Cirq and PyQuil are similar: Python + quantum. However, just below the surface we begin to see the differences. For example, PyQuil programs are succinct, while Cirq programs are verbose. In return, PyQuil programs are less flexible, while Cirq programs are more under programmer control. The homework is to program and run well-known algorithms in Cirq.

Purpose: Persuade you that you can get this to work.

Preview:

1. We will cover the Cirq view of qubits, moments, operations, and circuits.
2. We will show how to define new gates in Cirq.
3. We will cover such advanced topics as parameterization, noise, and optimization.

Transition to Body: Let us begin with the basics.

Main Point 1: We will cover the Cirq view of qubits, moments, operations, and circuits.

[Cirq has multiple ways of building circuits]
One way is first write a list of uses of gates and then map the list to a circuit. Another way is to use Python generators and yield statements.

[Cirq has syntax for running a simulation once or multiple times]
The basic expression for running a simulation of a circuit is `simulator.run(circuit)`. This can be enhanced with syntax for repeating the simulation.

[Cirq gives access to intermediate and final results]
We can access both internal data structures and the final results of a simulation. We might do that for the purposes of debugging and for doing statistics on the output.

Transition to MP2: Now let us go beyond what is hardwired in Cirq.

Main Point 2: We will show how to define new gates in Cirq.

[We can define a new gate as a class]
We can define a class of gates that inherits from an existing class of gates.

[We have to define how it works and how to display it in a diagram]
We must define the method `_unitary_` that defines how the gate works. We must also define the method `__str__` that defines how to display the gate in a diagram.

[We can also specify how to decompose it into a product of gates]
Cirq does rather little to figure out whether a given gate can be compiled to a native gate set. Instead, for each new gate, the programmer must give a specification of how to do such a translation. This is done by defining the method `_decompose_`. Cirq does no check of whether the decomposition is correct.

Transition to MP3: Finally let us talk about aspects of Cirq that are different from PyQuil.

Main Point 3: We will cover such advanced topics as parameterization, noise, and optimization.

[Paraterization uses symbolic values that are resolved at run time]
As often seen in programming, we may want to parameterize a circuit by a value that we can supply every time we run the circuit. This can be done in Cirq via use of symbolic values.

[Cirq supports a noise model that makes a simulator mimic a quantum computer]
Any quantum computer has noise, which means that gates and measurements are less than fully reliable. In contrast, simulators tend to execute gates and measurements in a fully reliable manner. In Cirq, we can make a simulation be more like running on quantum computer by adding noise. This is done by adding what Cirq calls a noise model. For example, one can add noise for each operation or one can add noise in every moment of a simulation.

[Cirq supports optimization-on-demand] The default of Cirq is to do no optimization of a circuit. However, the programmer can specify optimization on demand, via

```
XZOptimizer().optimize_circuit(circuit)
```

Transition to Close: So there you have it.

Review: Cirq has a syntax for circuits that is verbose yet also flexible, which is useful for tasks such as definition of new gates. Additionally, Cirq has advanced features that support both programming of large circuits and simulating in a way that mimics running on a quantum computer.

Strong finish: Cirq is developed and supported by a sizable team at Google. Research groups at University of Wisconsin and Duke University are working on making Cirq run on their own quantum computers. Those quantum computers are significantly different from Google's quantum computer, which uses superconducting qubits.

Call to action: Do you like Cirq or PyQuil better?