# Introduction to Quantum Programming and Semantics

Week 1: Semantics, quantum circuits

Chris Heunen

THE UNIVERSITY *of* EDINBURGH
informatics

## Semantics

Are these two programs the same?

$$P = (\texttt{if 1 = 1 then F else G})$$
$$Q = (\texttt{if 1 = 0 then F else F})$$

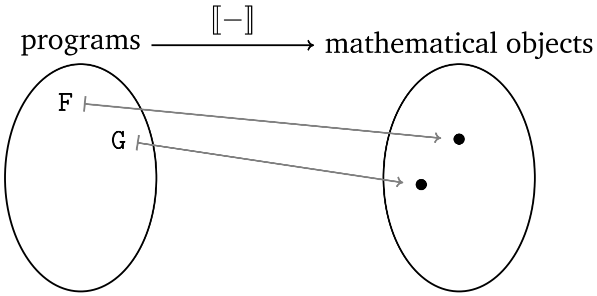
## Semantics

Are these two programs the same?

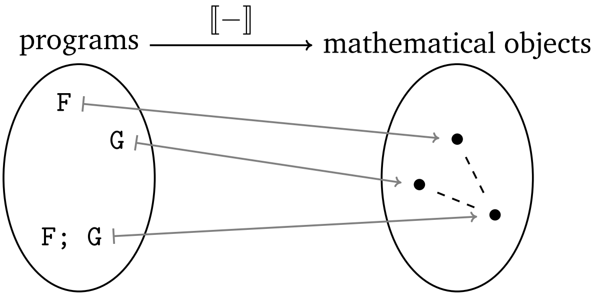
$$P = (\texttt{if 1 = 1 then F else G})$$
$$Q = (\texttt{if 1 = 0 then F else F})$$

- Different syntax
- Different operationally
- But denote same algorithm $[\![P]\!] = [\![Q]\!] = [\![\texttt{F}]\!]$

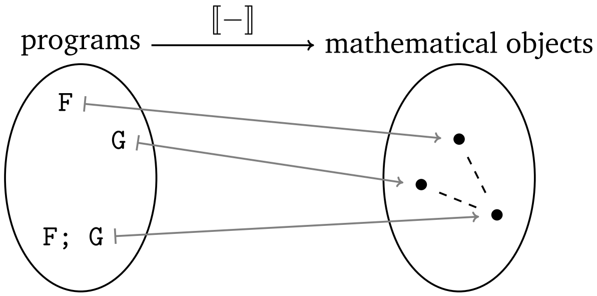
## Denotational semantics



## Denotational semantics

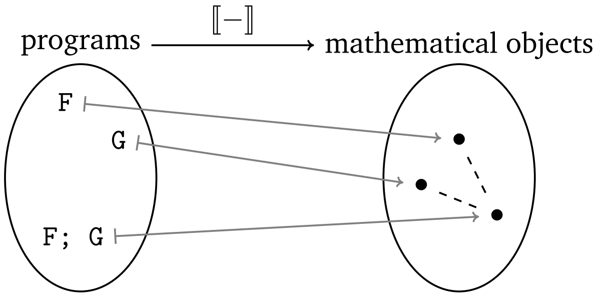


# Denotational semantics



- *Operational*: remember implementation details (efficiency)
- *Denotational*: see what program does conceptually (correctness)

## Denotational semantics

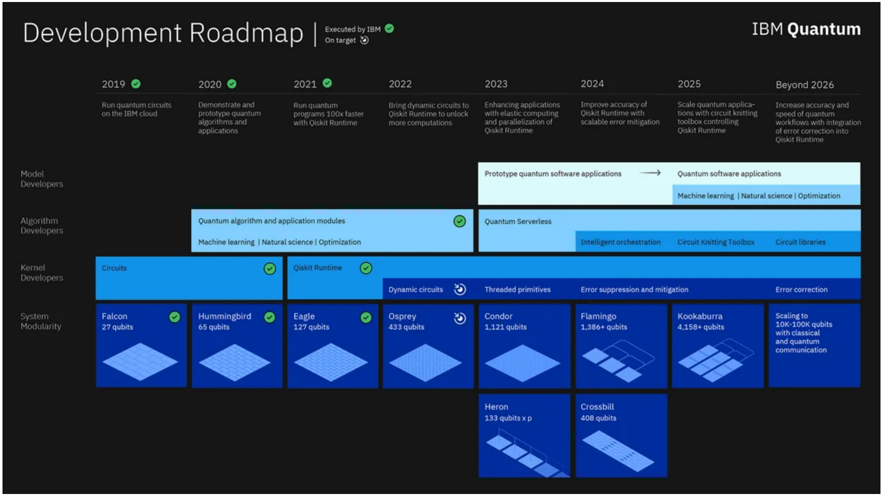


- *Operational*: remember implementation details (efficiency)
- *Denotational*: see what program does conceptually (correctness)

Motivation:

- Ground programmer's unspoken intuitions
- Justify/refute/suggest program transformations
- Understand programming through mathematics

## Quantum technology



Development Roadmap | Executed by IBM ✓ On target

IBM **Quantum**

2019 ✓ Run quantum circuits on the IBM cloud

2020 ✓ Demonstrate and prototype quantum algorithms and applications

2021 ✓ Run quantum programs 100x faster with Qiskit Runtime

2022 Bring dynamic circuits to Qiskit Runtime to unlock more computations

2023 Enhancing applications with elastic computing and parallelization of Qiskit Runtime

2024 Improve accuracy of Qiskit Runtime with scalable error mitigation

2025 Scale quantum applications with circuit knitting toolbox controlling Qiskit Runtime

Beyond 2026 Increase accuracy and speed of quantum workflows with integration of error correction into Qiskit Runtime

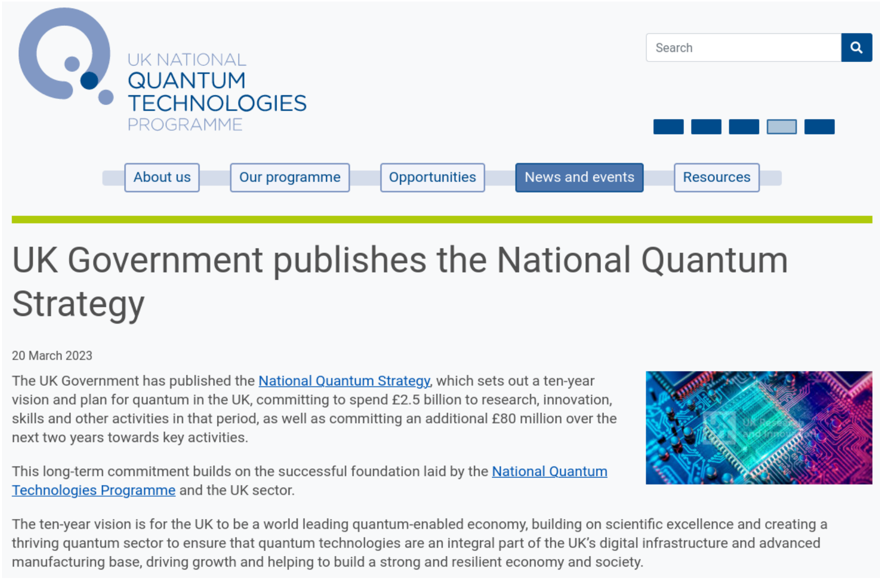
Model Developers: Prototype quantum software applications → Quantum software applications; Machine learning | Natural science | Optimization

Algorithm Developers: Quantum algorithm and application modules ✓; Machine learning | Natural science | Optimization; Quantum Serverless; Intelligent orchestration; Circuit Knitting Toolbox; Circuit libraries

Kernel Developers: Circuits ✓; Qiskit Runtime ✓; Dynamic circuits; Threaded primitives; Error suppression and mitigation; Error correction

System Modularity: Falcon 27 qubits ✓; Hummingbird 65 qubits ✓; Eagle 127 qubits ✓; Osprey 433 qubits; Condor 1,121 qubits; Flamingo 1,386+ qubits; Kookaburra 4,158+ qubits; Scaling to 10K-100K qubits with classical and quantum communication; Heron 133 qubits x p; Crossbill 408 qubits

## Quantum ecosystem



UK NATIONAL QUANTUM TECHNOLOGIES PROGRAMME

Search

About us | Our programme | Opportunities | News and events | Resources

UK Government publishes the National Quantum Strategy

20 March 2023

The UK Government has published the National Quantum Strategy, which sets out a ten-year vision and plan for quantum in the UK, committing to spend £2.5 billion to research, innovation, skills and other activities in that period, as well as committing an additional £80 million over the next two years towards key activities.

This long-term commitment builds on the successful foundation laid by the National Quantum Technologies Programme and the UK sector.

The ten-year vision is for the UK to be a world leading quantum-enabled economy, building on scientific excellence and creating a thriving quantum sector to ensure that quantum technologies are an integral part of the UK's digital infrastructure and advanced manufacturing base, driving growth and helping to build a strong and resilient economy and society.

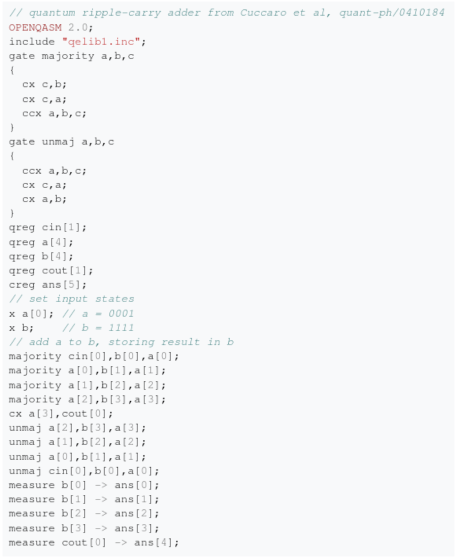
## Programming quantum computers

- What if $P, Q$ executables instead of source code? Black box.
  But can still analyse *information flow*
- Empirical method: know how quantum theory works, but why?
- Cannot copy or delete, how to handle recursion?

## Programming quantum computers

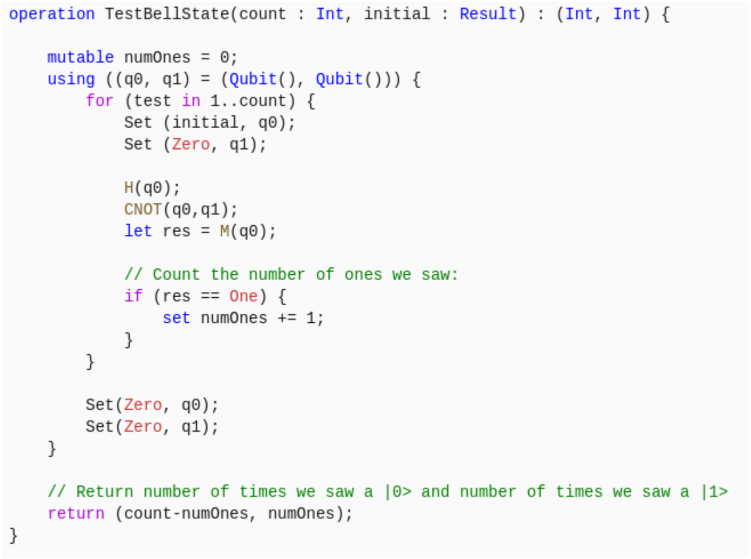
- What if $P, Q$ executables instead of source code? Black box. But can still analyse *information flow*
- Empirical method: know how quantum theory works, but why?
- Cannot copy or delete, how to handle recursion?
- Investigate semantics to design good programming language
- "Semantics = programming language"

## Need for abstraction



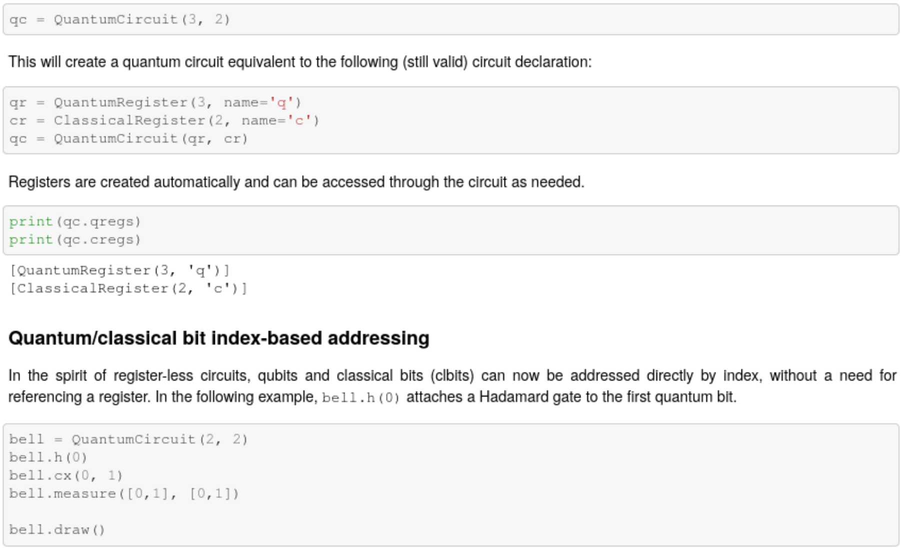
```
// quantum ripple-carry adder from Cuccaro et al, quant-ph/0410184
OPENQASM 2.0;
include "qelib1.inc";
gate majority a,b,c
{
  cx c,b;
  cx c,a;
  ccx a,b,c;
}
gate unmaj a,b,c
{
  ccx a,b,c;
  cx c,a;
  cx a,b;
}
qreg cin[1];
qreg a[4];
qreg b[4];
qreg cout[1];
creg ans[5];
// set input states
x a[0]; // a = 0001
x b;    // b = 1111
// add a to b, storing result in b
majority cin[0],b[0],a[0];
majority a[0],b[1],a[1];
majority a[1],b[2],a[2];
majority a[2],b[3],a[3];
cx a[3],cout[0];
unmaj a[2],b[3],a[3];
unmaj a[1],b[2],a[2];
unmaj a[0],b[1],a[1];
unmaj cin[0],b[0],a[0];
measure b[0] -> ans[0];
measure b[1] -> ans[1];
measure b[2] -> ans[2];
measure b[3] -> ans[3];
measure cout[0] -> ans[4];
```

## Need for abstraction: Microsoft Q#



```
operation TestBellState(count : Int, initial : Result) : (Int, Int) {

    mutable numOnes = 0;
    using ((q0, q1) = (Qubit(), Qubit())) {
        for (test in 1..count) {
            Set (initial, q0);
            Set (Zero, q1);

            H(q0);
            CNOT(q0,q1);
            let res = M(q0);

            // Count the number of ones we saw:
            if (res == One) {
                set numOnes += 1;
            }
        }

        Set(Zero, q0);
        Set(Zero, q1);
    }

    // Return number of times we saw a |0> and number of times we saw a |1>
    return (count-numOnes, numOnes);
}
```

## Need for abstraction: IBM Qiskit



```
qc = QuantumCircuit(3, 2)
```

This will create a quantum circuit equivalent to the following (still valid) circuit declaration:

```
qr = QuantumRegister(3, name='q')
cr = ClassicalRegister(2, name='c')
qc = QuantumCircuit(qr, cr)
```

Registers are created automatically and can be accessed through the circuit as needed.

```
print(qc.qregs)
print(qc.cregs)
```

```
[QuantumRegister(3, 'q')]
[ClassicalRegister(2, 'c')]
```

### Quantum/classical bit index-based addressing

In the spirit of register-less circuits, qubits and classical bits (clbits) can now be addressed directly by index, without a need for referencing a register. In the following example, `bell.h(0)` attaches a Hadamard gate to the first quantum bit.

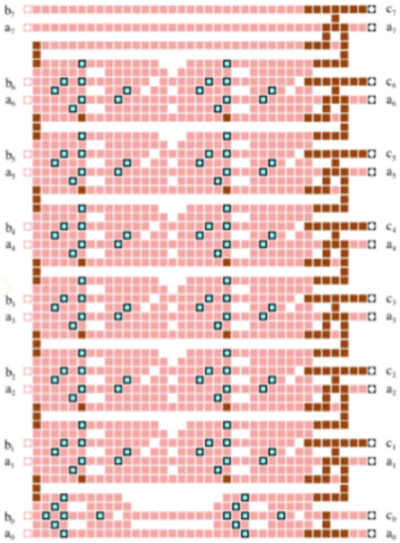
```
bell = QuantumCircuit(2, 2)
bell.h(0)
bell.cx(0, 1)
bell.measure([0,1], [0,1])

bell.draw()
```

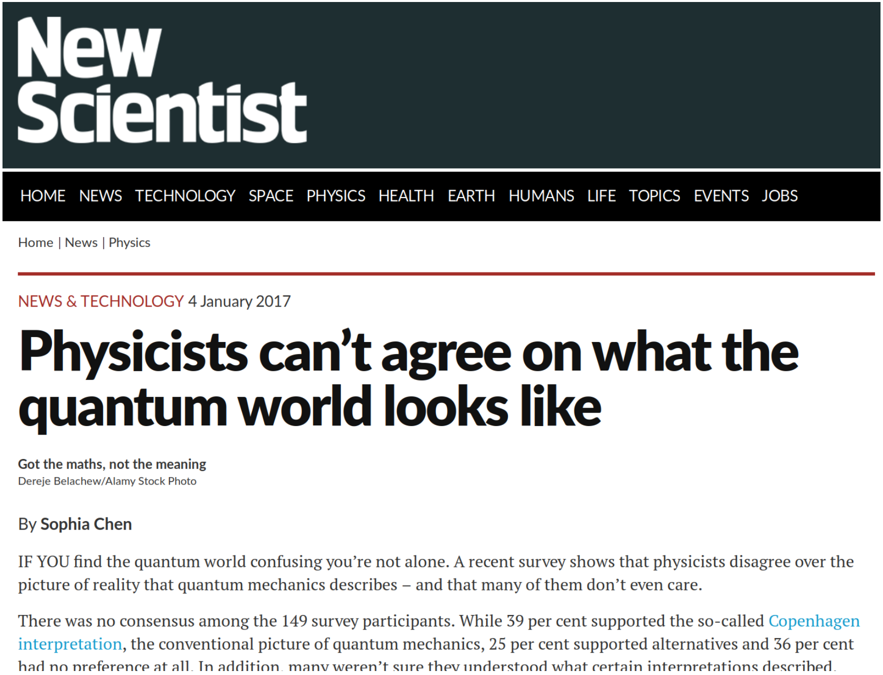
## Need for abstraction: Quipper

```
qft' :: [Qubit] -> Circ [Qubit]
qft' [] = return []
qft' [x] = do
  hadamard x
  return [x]
qft' (x:xs) = do
  xs' <- qft' xs
  xs'' <- rotations x xs' (length xs')
  x' <- hadamard x
  return (x':xs'')
 where
   rotations :: Qubit -> [Qubit] -> Int -> Circ [Qubit]
   rotations _ [] _ = return []
   rotations c (q:qs) n = do
     qs' <- rotations c qs n
     let m = ((n + 1) - length qs)
     q' <- rGate m q `controlled` c
     return (q':qs')
```

# Need for abstraction

8-bit adder, dimension $\sim 2^{1764}$





New Scientist
HOME NEWS TECHNOLOGY SPACE PHYSICS HEALTH EARTH HUMANS LIFE TOPICS EVENTS JOBS
Home | News | Physics

NEWS & TECHNOLOGY 4 January 2017

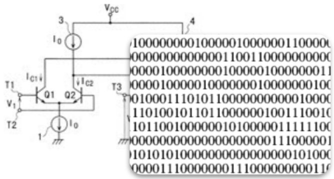
# Physicists can't agree on what the quantum world looks like

**Got the maths, not the meaning**
Dereje Belachew/Alamy Stock Photo

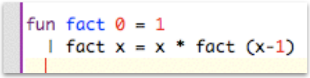
By **Sophia Chen**

IF YOU find the quantum world confusing you're not alone. A recent survey shows that physicists disagree over the picture of reality that quantum mechanics describes – and that many of them don't even care.

There was no consensus among the 149 survey participants. While 39 per cent supported the so-called Copenhagen interpretation, the conventional picture of quantum mechanics, 25 per cent supported alternatives and 36 per cent had no preference at all. In addition, many weren't sure they understood what certain interpretations described.



$$\lambda x.\lambda y.\lambda z.xz(yz)$$



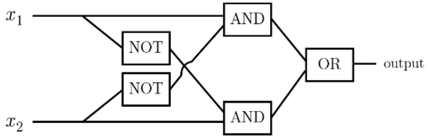
```
fun fact 0 = 1
  | fact x = x * fact (x-1)
```

Hilbert space, unitary transforms, self-adjoint operators....

?

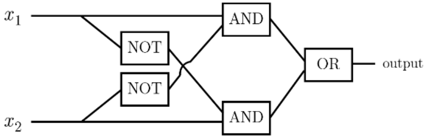
## Ways to look at computation

- function $\{0,1\}^m \to \{0,1\}^n$
- truth table / $m$-by-$n$ matrix with entries in $\{0,1\}$
- Boolean circuit



## Ways to look at computation

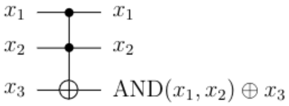
- function $\{0,1\}^m \to \{0,1\}^n$
- truth table / $m$-by-$n$ matrix with entries in $\{0,1\}$
- Boolean circuit



AND, OR, and NOT are universal (as are other gate sets)

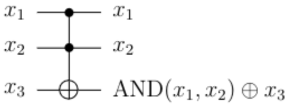
## Reversible computation

- function is bijective
- matrix is invertible
- circuit of CCNOT gates

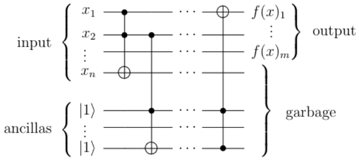


# Reversible computation

- function is bijective
- matrix is invertible
- circuit of CCNOT gates



CCNOT is universal



## Quantum computation

- *linear* function $\mathbb{C}^{2^m} \longrightarrow \mathbb{C}^{2^n}$
- *unitary* $m$-by-$n$ matrix of *complex numbers*
- *quantum circuit* of CCNOT and $H$ gates

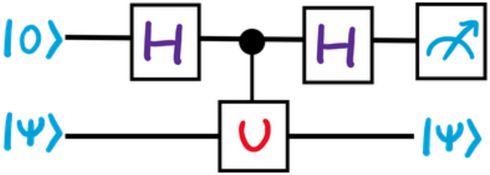
## Quantum computation

- *linear* function $\mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$
- *unitary* $m$-by-$n$ matrix of *complex numbers*
- *quantum circuit* of CCNOT and $H$ gates

CCNOT and $H$ are *approximately* universal

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

## Phase kickback



## Quantum lock

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

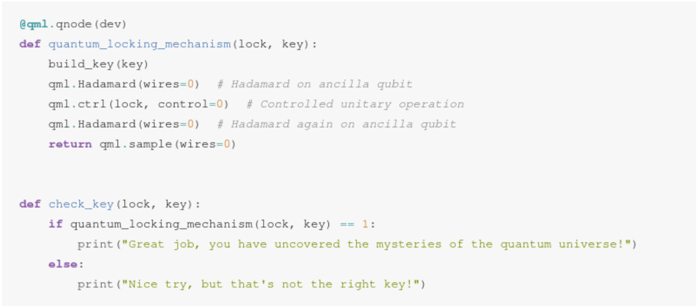
$$U|key\rangle = -|key\rangle$$

## Quantum lock

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

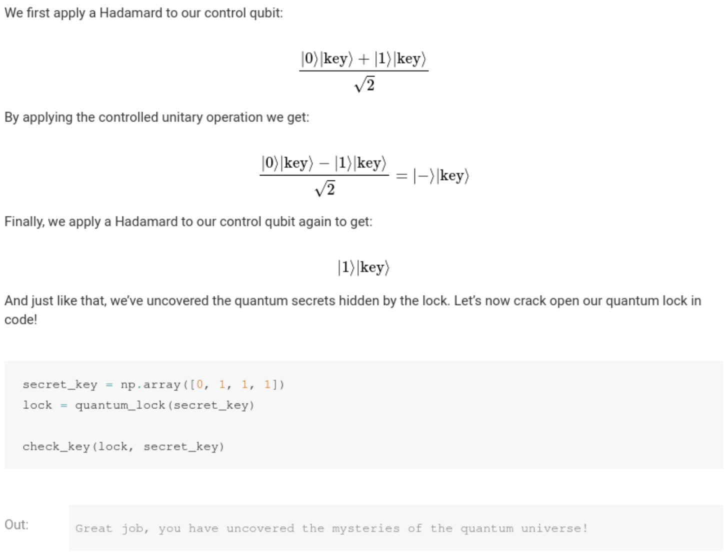
$$U|key\rangle = -|key\rangle$$

In PennyLane:



```python
@qml.qnode(dev)
def quantum_locking_mechanism(lock, key):
    build_key(key)
    qml.Hadamard(wires=0)  # Hadamard on ancilla qubit
    qml.ctrl(lock, control=0)  # Controlled unitary operation
    qml.Hadamard(wires=0)  # Hadamard again on ancilla qubit
    return qml.sample(wires=0)


def check_key(lock, key):
    if quantum_locking_mechanism(lock, key) == 1:
        print("Great job, you have uncovered the mysteries of the quantum universe!")
    else:
        print("Nice try, but that's not the right key!")
```

## Correct key



We first apply a Hadamard to our control qubit:

$$\frac{|0\rangle|\text{key}\rangle + |1\rangle|\text{key}\rangle}{\sqrt{2}}$$

By applying the controlled unitary operation we get:

$$\frac{|0\rangle|\text{key}\rangle - |1\rangle|\text{key}\rangle}{\sqrt{2}} = |-\rangle|\text{key}\rangle$$

Finally, we apply a Hadamard to our control qubit again to get:

$$|1\rangle|\text{key}\rangle$$

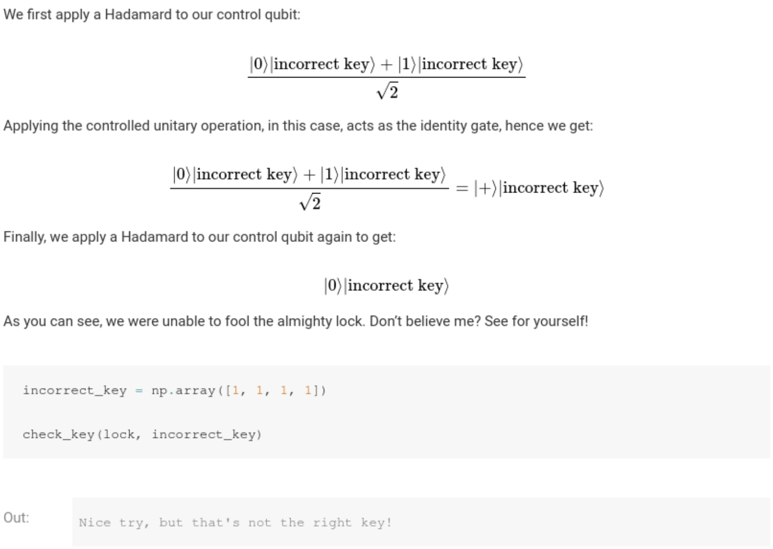
And just like that, we've uncovered the quantum secrets hidden by the lock. Let's now crack open our quantum lock in code!

```
secret_key = np.array([0, 1, 1, 1])
lock = quantum_lock(secret_key)

check_key(lock, secret_key)
```

Out:

```
Great job, you have uncovered the mysteries of the quantum universe!
```

## Incorrect key



We first apply a Hadamard to our control qubit:

$$\frac{|0\rangle|\text{incorrect key}\rangle + |1\rangle|\text{incorrect key}\rangle}{\sqrt{2}}$$

Applying the controlled unitary operation, in this case, acts as the identity gate, hence we get:

$$\frac{|0\rangle|\text{incorrect key}\rangle + |1\rangle|\text{incorrect key}\rangle}{\sqrt{2}} = |+\rangle|\text{incorrect key}\rangle$$

Finally, we apply a Hadamard to our control qubit again to get:

$$|0\rangle|\text{incorrect key}\rangle$$

As you can see, we were unable to fool the almighty lock. Don't believe me? See for yourself!

```
incorrect_key = np.array([1, 1, 1, 1])

check_key(lock, incorrect_key)
```

Out:

```
Nice try, but that's not the right key!
```

## Summary

- Denotational semantics preserves structure: not implementation
- Good programming language flows from semantics:
  "form follows function"
- Quantum computation: circuit vs matrix vs linear function
- Example: phase kickback