Deutsch's Algorithm and Logical Operations: A Graduate-Level Quantum Computing Laboratory

Your Name

April 3, 2025

Contents

Chapter 1

Introduction and Preliminaries

1.1 Abstract Algebraic and Logical Foundations

In these notes we adopt an abstract algebra viewpoint. A *vector space* is regarded as an abelian group endowed with an action of a field. In our case the underlying field is $\mathbb{C}$, and the additive structure is that of an abelian group. In addition, we review fundamental Boolean operations that are used in describing the quantum oracle.

Boolean Operations

Let $A, B \in \{0, 1\}$. We recall:

- **Negation (NOT):**

  $$\overline{A} \quad \text{or} \quad A' \quad (\text{also denoted } \sim A)$$

  with truth table:

A	$\overline{A}$
0	1
1	0

- **Conjunction (AND):**

  $$A \wedge B \quad \text{or simply } AB,$$

  with

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

- **Disjunction (OR):**

  $$A \vee B \quad \text{or } A + B,$$

  with

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

- **Exclusive OR (XOR):**

  $$A \oplus B,$$

defined as modulo-2 addition:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

In the context of quantum computing, the XOR operation is implemented by the controlled-NOT gate (CNOT) and is equivalent to the Kronecker (tensor) product in its matrix form.

1.2 The Oracle in Deutsch's Algorithm

Consider a function

$$f : \{0, 1\} \to \{0, 1\},$$

with the promise that f is either *constant* (i.e., $f(0) = f(1)$) or *balanced* (i.e., $f(0) \neq f(1)$). In the quantum setting, we encode this function into a unitary operator (oracle) U_f defined by:

$$U_f \colon |x, y\rangle \mapsto |x,\, y \oplus f(x)\rangle \,.$$

Notice that when $f(x) = 0$, the oracle acts as the identity on the target qubit, and when $f(x) = 1$, it acts as the NOT operator, i.e.,

$$U_f \,|x, y\rangle = \begin{cases} |x, y\rangle \,, & f(x) = 0, \\ |x, \overline{y}\rangle \,, & f(x) = 1. \end{cases}$$

This formulation guarantees the reversibility required for unitary evolution.

Chapter 2

Deutsch's Algorithm: Quantum Circuit and Detailed Analysis

2.1 Quantum Circuit Implementation

The Deutsch algorithm determines the nature of f with one query to the oracle by exploiting quantum superposition and interference.

Circuit Overview

The quantum circuit consists of:

Step 1: Initialization: Prepare the two-qubit state

$$|\psi_0\rangle = |0\rangle \otimes |1\rangle .$$

Step 2: Superposition via Hadamard: Apply the Hadamard gate H to each qubit, where

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

The state becomes

$$|\psi_1\rangle = \frac{1}{2}\left[(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)\right].$$

Step 3: Oracle Query: Apply U_f, resulting in the state

$$|\psi_2\rangle = \frac{1}{2}\Big[|0,\, 0 \oplus f(0)\rangle - |0,\, 1 \oplus f(0)\rangle + |1,\, 0 \oplus f(1)\rangle - |1,\, 1 \oplus f(1)\rangle\Big].$$

Step 4: Interference via Hadamard on the Control Qubit: Apply $H \otimes I$ to yield a state in which the amplitude of $|0\rangle$ on the control qubit is proportional to

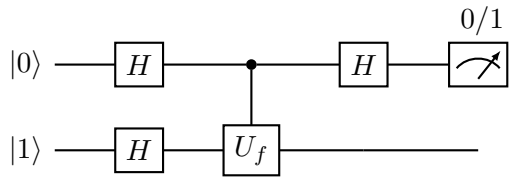
$$(-1)^{f(0)} + (-1)^{f(1)}.$$

Step 5: Measurement: Measuring the control qubit then reveals:

$$\begin{cases} |0\rangle & \text{if } f(0) = f(1) \quad \text{(constant)}, \\ |1\rangle & \text{if } f(0) \neq f(1) \quad \text{(balanced)}. \end{cases}$$

Circuit Diagram

For illustration, the following quantum circuit (using `quantikz`) represents the above steps:



2.2 Detailed State Evolution

After Hadamard Gates

Starting from

$$|\psi_0\rangle = |0\rangle \otimes |1\rangle\,,$$

the Hadamard transform gives:

$$|\psi_1\rangle = \frac{1}{2}\Big[|0,0\rangle - |0,1\rangle + |1,0\rangle - |1,1\rangle\Big].$$

After Oracle Application

Using

$$U_f\,|x,y\rangle = |x,\, y \oplus f(x)\rangle\,,$$

we have:

$$|\psi_2\rangle = \frac{1}{2}\Big[|0,f(0)\rangle - |0,1-f(0)\rangle + |1,f(1)\rangle - |1,1-f(1)\rangle\Big].$$

After Final Hadamard on the First Qubit

Applying H to the first qubit, we obtain interference patterns such that the amplitude on $|0\rangle$ is proportional to

$$(-1)^{f(0)} + (-1)^{f(1)}.$$

Thus, measurement of the first qubit distinguishes the two cases:

$$\text{If } f(0) = f(1): \quad (-1)^{f(0)} + (-1)^{f(1)} = \pm 2, \quad \text{yielding outcome } |0\rangle\,,$$

$$\text{If } f(0) \neq f(1): \quad (-1)^{f(0)} + (-1)^{f(1)} = 0, \quad \text{yielding outcome } |1\rangle\,.$$

2.3 Uncomputation and Reversibility

An important aspect of the quantum oracle is that it is a reversible (unitary) operator. In some implementations the operation U_f is "uncomputed" (i.e., reversed) to eliminate ancillary information. For example, one may show that

$$U_f\,|x,y\rangle = |x, y \oplus f(x)\rangle$$

can be inverted by applying the same operator U_f a second time:

$$U_f^2 = \mathbb{I},$$

since modulo-2 addition is its own inverse.

Chapter 3

Lab Implementation and Programming Aspects

3.1 Overview of the Laboratory

In this laboratory we implement Deutsch's algorithm on a quantum circuit simulator (using, e.g., Qiskit). In addition, we review how to use Boolean logic operators in Python, including:

- `not` (logical NOT),
- `and` (logical AND),
- `or` (logical OR),
- `xor` (exclusive OR, which in Python can be implemented via the caret operator `^`).

3.2 Python Implementation Highlights

For example, when defining the oracle in Qiskit:

```
def Uf(qc, f):
"""
Implements the oracle U_f:
U_f|x,y> = |x, y xor f(x)>
"""
# Apply CNOT if f(x)==1; otherwise, do nothing.
for qubit in qc.qubits:
if f(qubit):  # pseudo-code: depends on implementation of f
qc.cx(control, target)
```

Additionally, one may fix the seed of the random number generator (using `numpy.random.seed()`) to ensure reproducibility of random choices in the lab.

3.3 Slide Outline

Below is a suggested slide sequence for the lecture:

Slide 1: Deutsch Algorithm Implementation Overview.

Slide 2: Statement of Deutsch Algorithm.

Slide 3: Boolean Logic: NOT Operator.

Slide 4: Boolean Logic: AND Operator.

Slide 5: Boolean Logic: OR Operator.

Slide 6: Boolean Logic: XOR Operator.

Slide 7: Combined Logic Operations: NOT, AND, OR, XOR.

Slide 8: [Additional slides as needed for details.]

Slide 9: Oracle Circuit Uncomputation.

Slide 10: Python Random Seed and Control Flow (`if` statements).

Slide 11: Use of `QuantumCircuit.compose()`.

Slide 12: Automatic creation of classical registers via `QuantumCircuit.measure_all()`.

Slide 13: Deutsch Algorithm Recap.

Slide 14: Criteria for Constant versus Balanced Functions.

Slide 15: Examples using Python dictionaries (`dict` class) and formatted strings.

Slide 16: Uploading and downloading `.ipynb` files on Colab.

Slide 17: Final submission instructions.

Chapter 4

Exercises

Exercise 1: **State Decomposition:** Express the two-qubit state at the point of the oracle (or after the red-dashed stage in the circuit) as a linear combination of the computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$.

Exercise 2: **Measurement Probabilities:** Assuming that $f(0) = f(1) = 1$, compute the probabilities of obtaining measurement outcomes 0 and 1 for the control qubit.

Exercise 3: **Oracle Uncomputation:** Show that the operator U_f, defined by $U_f |x, y\rangle = |x, y \oplus f(x)\rangle$, is its own inverse.

Chapter 5

Conclusion

We have rigorously described Deutsch's algorithm from an abstract algebra and quantum computing viewpoint. The discussion has spanned Boolean logic, the algebraic structure of vector spaces, quantum circuit implementation, and the subtleties of oracle uncomputation. This exposition is intended to serve as a detailed reference for both theoretical understanding and practical implementation in a laboratory setting.

References

[1] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information.*

[2] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proc. R. Soc. Lond. A* **400**, 97–117 (1985).

[3] P. Shor, *Introduction to Quantum Algorithms.*

[4] S. Lang, *Algebra.*