The quantum circuit model

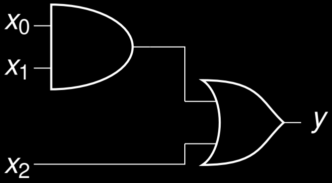
Massimiliano Incudini

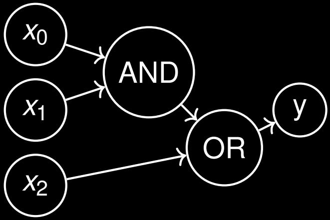
October 6, 2024

Circuit model

Boolean circuits

A Boolean circuit is a directed acyclic graph whose vertices are either inputs, outputs, or computational nodes representing the logical gates AND, OR, NOT.





Boolean functions

What can we use the Boolean circuit for?

Boolean functions

What can we use the Boolean circuit for?

Each Boolean circuit with n inputs computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$.

Boolean functions

What can we use the Boolean circuit for?

Each Boolean circuit with n inputs computes a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$.

Which Boolean functions can we compute?

Boolean functions

What can we use the Boolean circuit for?

Each Boolean circuit with n inputs computes a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$.

Which Boolean functions can we compute?

You can compute *any* Boolean function f using a circuit composed of AND, OR, and NOT gates.

Decision problem

Let $L \subseteq \{0, 1\}^*$ be a decision problem. Each string $x \in \{0, 1\}^*$ is an instance of such a problem.

Decision problem

Let $L \subseteq \{0,1\}^*$ be a decision problem. Each string $x \in \{0,1\}^*$ is an instance of such a problem.

Can we use the circuit framework to solve a decision problem?

Decision problem

Let $L \subseteq \{0,1\}^*$ be a decision problem. Each string $x \in \{0,1\}^*$ is an instance of such a problem.

Can we use the circuit framework to solve a decision problem?

Tentative:

A *circuit family* $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ is a sequence of circuits, one for each input size n.

$\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ solves L if

- for all $x \in L, |x| = n$ we have $C_n(x) = 1$
- for all $x \notin L, |x| = n$ we have $C_n(x) = 0$

Uniformity

Problem: The class of decision problems solvable by the circuit family model is huge.

Uniformity

Problem: The class of decision problems solvable by the circuit family model is huge.

Root of the problem: We can have a different circuit per input.

Uniformity

Problem: The class of decision problems solvable by the circuit family model is huge.

Root of the problem: We can have a different circuit per input.

What if we have a single, finite program that can generate all the circuits?

Uniformity

Problem: The class of decision problems solvable by the circuit family model is huge.

Root of the problem: We can have a different circuit per input.

What if we have a single, finite program that can generate all the circuits?

A circuit family $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ is *uniformly polynomial* if there exists a Turing machine T that for input n gives a description of C_n in $\mathcal{O}(\text{poly}\log n)$ space.

P

A decision problem L is in the complexity class P if there exists a uniformly polynomial circuit family $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ such that

- for every $x \in L$ with $|x| = n$ we have $C_n(x) = 1$
- for every $x \notin L$ with $|x| = n$ we have $C_n(x) = 0$

BPP

A randomized circuit family $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ is a sequence of circuits, one for each input size n, such that each circuit is provided with $\mathcal{O}(\text{poly}\, n)$ random bits in addition to the input.

BPP

A randomized circuit family $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ is a sequence of circuits, one for each input size n, such that each circuit is provided with $\mathcal{O}(\text{poly}\, n)$ random bits in addition to the input.

A decision problem L is in the complexity class BPP if there exists a uniformly polynomial randomized circuit family $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ such that

- for all $x \in L, |x| = n$ we have $Pr[C_n(x) = 1] \geq 2/3$;
- for all $x \notin L, |x| = n$ we have $Pr[C_n(x) = 0] \leq 1/3$.

BPP

A randomized circuit family $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ is a sequence of circuits, one for each input size n, such that each circuit is provided with $\mathcal{O}(\text{poly}\ n)$ random bits in addition to the input.

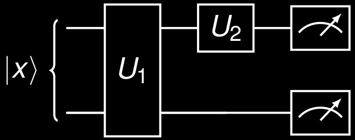
A decision problem L is in the complexity class BPP if there exists a uniformly polynomial randomized circuit family $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ such that

- for all $x \in L, |x| = n$ we have $Pr[C_n(x) = 1] \geq 2/3$;
- for all $x \notin L, |x| = n$ we have $Pr[C_n(x) = 0] \leq 1/3$.

We can improve the value of this threshold.

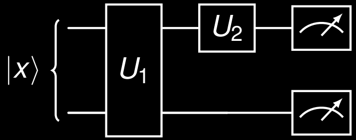
Quantum circuits

A quantum circuit:



Quantum circuits

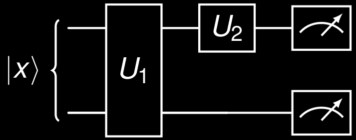
A quantum circuit:



- Start in the computational basis state $|x\rangle$

Quantum circuits

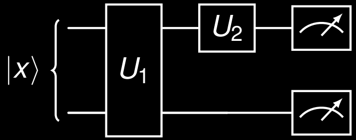
A quantum circuit:



- Start in the computational basis state $|x\rangle$
- Reversibility: the number of fan-in and fan-out is the same in each layer

Quantum circuits

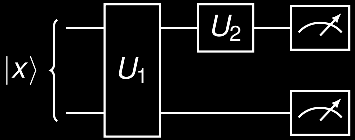
A quantum circuit:



- Start in the computational basis state $|x\rangle$
- Reversibility: the number of fan-in and fan-out is the same in each layer
- Parallel operations are composed via the tensor product

Quantum circuits

A quantum circuit:



- Start in the computational basis state $|x\rangle$
- Reversibility: the number of fan-in and fan-out is the same in each layer
- Parallel operations are composed via the tensor product
- Sequential operations are composed by multiplication

Quantum gates

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{pmatrix}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \mathbb{I} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Universality (quantum model)

Which unitaries can I implement with a quantum circuit?

Universality (quantum model)

Which unitaries can I implement with a quantum circuit?

$$\downarrow$$

Every unitary U over n-qubit
can be implemented using a quantum circuit
composed of 1- and 2-qubit unitaries

Universality (quantum model)

Which unitaries can I implement with a quantum circuit?

$\downarrow$

Every unitary *U* over *n*-qubit
can be implemented using a quantum circuit
composed of 1- and 2-qubit unitaries

$\downarrow$

Every 1- and 2-qubit unitary
can be *approximately* implemented
using only gates from a small, finite set
e.g. $\mathcal{G} = \{\textsc{cnot}, H, T\}$

BQP

A decision problem $L \subseteq \{0,1\}^*$ is in the complexity class BQP if there exists a uniformly polynomial quantum circuit family $\mathcal{C} = \{Q_n\}_{n \in \mathbb{N}}$ such that

- for all $x \in L, |x| = n$ we have $Pr[Q_n(x) = 1] \geq 2/3$;
- for all $x \notin L, |x| = n$ we have $Pr[Q_n(x) = 0] \leq 1/3$.

Query model

Oracle

The query model differs substantially from the circuit model:

Oracle

The query model differs substantially from the circuit model:

1. We do not begin with a quantum state that depends on the input.

Oracle

The query model differs substantially from the circuit model:

1. We do not begin with a quantum state that depends on the input.
2. The input $x \in \{0,1\}^N$, with $N = 2^n$, is provided as a black-box oracle O_x:

$$O_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle .$$

Each use or call of the oracle is referred to as a *query*.

Oracle

The query model differs substantially from the circuit model:

1. We do not begin with a quantum state that depends on the input.
2. The input $x \in \{0,1\}^N$, with $N = 2^n$, is provided as a black-box oracle O_x:

   $$O_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle .$$

   Each use or call of the oracle is referred to as a *query*.
3. The quantum circuit interleaves the queries to the oracle with standard, non-query operations.

Motivation

It is difficult to prove lower bounds on the complexity of computing some functions over explicit input data.

In contrast, we can often demonstrate that many queries are required to compute some given function of the black-box input.

Phase Oracle

The input $x \in \{0,1\}^N$, with $N = 2^n$, is provided as a black-box *phase oracle* $O_x^{\pm}$ if:

$$O_x^{\pm} |i\rangle = (-1)^{x_i} |i\rangle .$$

Equivalence between oracle formats

The circuit implementing the phase oracle given a traditional oracle is $O_x^{\pm} = (\mathbb{I} \otimes H) O_x (\mathbb{I} \otimes HX)$.

[whiteboard]

Whiteboard

$$
\begin{aligned}
2^{-1/2} O_x \left|i\right\rangle (\left|0\right\rangle - \left|1\right\rangle) &= 2^{-1/2} \left|i\right\rangle (\left|0 \oplus x_i\right\rangle - \left|1 \oplus x_i\right\rangle) \\
&= 2^{-1/2} \begin{cases} \left|i\right\rangle (\left|0\right\rangle - \left|1\right\rangle), & x_i = 0 \\ \left|i\right\rangle (-1)(\left|0\right\rangle - \left|1\right\rangle), & x_i = 1 \end{cases} \\
&= 2^{-1/2} \begin{cases} \left|i\right\rangle (-1)^0(\left|0\right\rangle - \left|1\right\rangle), & x_i = 0 \\ \left|i\right\rangle (-1)^1(\left|0\right\rangle - \left|1\right\rangle), & x_i = 1 \end{cases} \\
&= 2^{-1/2}(-1)^{x_i} \left|i\right\rangle (\left|0\right\rangle - \left|1\right\rangle)
\end{aligned}
$$

Deutsch-Jozsa algorithm

BALANCED-OR-CONSTANT problem

Input: oracle access O_f to a Boolean function $f : \{0,1\}^n \to \{0,1\}$

Promise: f is either constant or balanced ($|f^{-1}(0)| = |f^{-1}(1)|$).

Output: 0 if f is constant and 1 if f is balanced.

The classical query complexity for the BALANCED-OR-CONSTANT problem is $N/2 + 1 \in \mathcal{O}(2^n)$.

The classical query complexity for the BALANCED-OR-CONSTANT problem is $N/2 + 1 \in \mathcal{O}(2^n)$.

The quantum query complexity for the BALANCED-OR-CONSTANT problem is 1. This is achieved using the DEUTSCH-JOZSA algorithm.

The classical query complexity for the BALANCED-OR-CONSTANT problem is $N/2 + 1 \in \mathcal{O}(2^n)$.

The quantum query complexity for the BALANCED-OR-CONSTANT problem is 1. This is achieved using the DEUTSCH-JOZSA algorithm.

[whiteboard]

Whiteboard

$$|\psi_0\rangle = |0^n\rangle$$

$$|\psi_1\rangle = H^{\otimes n}|0^n\rangle = 2^{-n/2}\sum_{i\in\{0,1\}^n}|i\rangle$$

$$|\psi_2\rangle = O_f^{\pm}\left(2^{-n/2}\sum_{i\in\{0,1\}^n}|i\rangle\right) = 2^{-n/2}\sum_{i\in\{0,1\}^n}(-1)^{f(i)}|i\rangle$$

Whiteboard

$$
\begin{aligned}
|\psi_3\rangle &= H^{\otimes n}\left(2^{-n/2}\sum_{i\in\{0,1\}^n}(-1)^{f(i)}\,|i\rangle\right)\\
&= 2^{-n/2}\sum_{i\in\{0,1\}^n}(-1)^{f(i)}H^{\otimes n}\,|i\rangle\\
&= 2^{-n}\sum_{i\in\{0,1\}^n}(-1)^{f(i)}\sum_{j\in\{0,1\}^n}(-1)^{i\cdot j}\,|j\rangle
\end{aligned}
$$

Here,

$$
\begin{aligned}
H^{\otimes n}\,|i\rangle &= (H\,|i_0\rangle)\otimes\cdots\otimes(H\,|i_{n-1}\rangle)\\
&= \frac{|0\rangle+(-1)^{i_0}\,|1\rangle}{\sqrt{2}}\otimes\cdots\otimes\frac{|0\rangle+(-1)^{i_{n-1}}\,|1\rangle}{\sqrt{2}}\\
&= \frac{1}{\sqrt{2}^n}\sum_{j\in\{0,1\}^n}(-1)^{i\cdot j}\,|j\rangle
\end{aligned}
$$

Whiteboard

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{f(i)} (-1)^{i \cdot 0} |0\rangle + \cdot ...$$

$$= \frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{f(i)} |0\rangle + \cdot ...$$

Here,

$$\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{f(i)} = 0 \text{ if } f \text{ is balanced}$$

$$\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{f(i)} = 1 \text{ if } f \text{ is constant one}$$

$$\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{f(i)} = -1 \text{ if } f \text{ is constant zero}$$

Note: If we allow a small constant probability of error, the BALANCED-OR-CONSTANT problem can be solved efficiently with a constant number of queries.

Bernstein-Vazirani algorithm

SECRET-STRING problem

Input: oracle access O_f to a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

Promise: $f(x) = (s \cdot x) \bmod 2$

Output: the n-bit string s

The classical query complexity for the SECRET-STRING problem is n:

$$f(100\cdots0) = s_0$$
$$f(010\cdots0) = s_1$$
$$f(001\cdots0) = s_2$$
$$f(000\cdots1) = s_{n-1}$$

Each query returns one bit of information. Therefore, we cannot do better than $\mathcal{O}(n)$.

The quantum query complexity for the SECRET-STRING problem is 1, as demonstrated by the BERNSTEIN-VAZIRANI algorithm.

[whiteboard]

Whiteboard

$$|\psi_2\rangle = O_f^{\pm} H^{\otimes n} \left|0^n\right\rangle = 2^{-n/2} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot s} |x\rangle$$

Recall that $H^{\otimes n} |s\rangle = 2^{-n/2} \sum_{y \in \{0,1\}^n} (-1)^{s \cdot y} |y\rangle$.

To recover s we just need to invert $H^{\otimes n}$. After that, we get $|s\rangle$ in the computational basis.

Thank you!