

Quantum circuit


Ryuhei Mori


Tokyo Institute of Technology

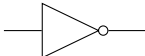
December 24, 2019

Boolean circuit

- Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$.
- **Boolean circuit** is a model of computation of a Boolean functions which consists of logic **gates**.

- AND gate: The AND gate symbol is a D-shaped logic symbol with two input lines on the left and one output line on the right.

- OR gate: The OR gate symbol is a symbol with a curved input side on the left and a pointed output side on the right, with two input lines and one output line.

- NOT gate: The NOT gate symbol is a triangular logic symbol with one input line on the left and one output line on the right, featuring a small circle (bubble) at the output tip.

Universality of {AND, OR, NOT}

Theorem

For *any* Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, there is a Boolean circuit with AND, OR and NOT gates computing f .

Proof.

The proof is an induction on n . Theorem is trivial for $n = 1$. Assume that Theorem holds for Boolean functions $n \leq k - 1$.

$$f(x_1, \dots, x_k) = (x_k \wedge f(x_1, \dots, x_{k-1}, 1)) \vee (\overline{x_k} \wedge f(x_1, \dots, x_{k-1}, 0)).$$



Size of Boolean circuits

Size of Boolean circuit := # of **gates** in the Boolean circuit.

Let $C(f)$ be a smallest size of Boolean circuit computing

$f: \{0, 1\}^n \rightarrow \{0, 1\}$.

Let $s(n) := \max_{f: \{0,1\}^n \rightarrow \{0,1\}} C(f)$.

$$f(x_1, \dots, x_n) = (x_n \wedge f(x_1, \dots, x_{n-1}, 1)) \vee (\overline{x_n} \wedge f(x_1, \dots, x_{n-1}, 0))$$

$$s(n) \leq c + 2s(n-1)$$

$$\frac{s(n)}{2^n} \leq \frac{c}{2^n} + \frac{s(n-1)}{2^{n-1}}$$

$$\leq c \left(\frac{1}{2^n} + \frac{1}{2^{n-1}} + \dots + \frac{1}{2} \right) + s(0) \leq c + s(0)$$

$$s(n) = O(2^n).$$

Lower bound of size of Boolean circuits

The number of Boolean functions with n variables is 2^{2^n} .

The number of Boolean circuits of size s is at most

$$\left(8 \binom{n+s}{2}\right)^s \leq (4(n+s)^2)^s.$$

This means

$$\begin{aligned} (4(n+s(n))^2)^{s(n)} &\geq 2^{2^n} \\ \iff s(n) \log(4(n+s(n))^2) &\geq 2^n \\ \implies s(n) &\geq \frac{2^n}{3n} \quad \text{for sufficiently large } n. \end{aligned}$$

In fact, $s(n) = \frac{2^n}{n}(1 + o(1))$.

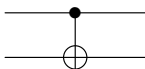
Quantum circuit

- **Quantum circuit** is a model of computation of Boolean functions which consists of **quantum gates**.

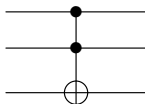
- Single qubit gate: X gate, Y gate, Z gate, H gate,

$$S := \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \text{ gate} \quad \text{---} \boxed{X} \text{---}$$

- Two qubit gate: CNOT gate



- Three qubit gate: Toffoli gate



“Classical computation” by a quantum circuit

Lemma

*For any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, there is a quantum circuit on $n + 1 + w$ qubits which consists of **X**, **CNOT** and **Toffoli** gates for U satisfying*

$$U |x\rangle |y\rangle |0\rangle^{\otimes w} = |x\rangle |y \oplus f(x)\rangle |0\rangle^{\otimes w}$$

for all $x \in \{0, 1\}^n$, $y \in \{0, 1\}$. Here, the number w of working qubits (ancilla) is at most $C(f) + 1$ and the number g of quantum gates is at most $2C(f) + 1$.

A sketch of a proof.

Translate a Boolean circuit to a quantum circuit.



Universality of a quantum circuit

Theorem (Universality of finite gate set)

For any unitary matrix $U \in L(\mathbb{C}^{2^n})$ and $\epsilon > 0$, there is a quantum circuit with $X, Y, Z, H, S, \text{CNOT}, \text{Toffoli}$ gates computing \tilde{U} satisfying $\|U - \tilde{U}\| < \epsilon$.

Proof.

In the next class.



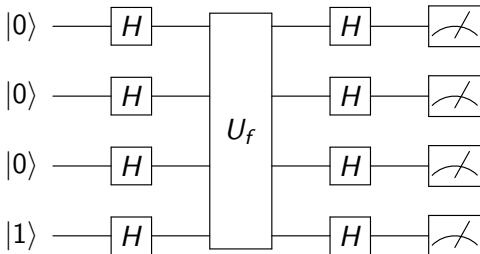
Oracle model

- Input is given by an oracle.
- Classical oracle: oracle gate $i \mapsto x_i$.
- Quantum oracle: quantum oracle gate $U |i\rangle |y\rangle = |i\rangle |y \oplus x_i\rangle$.
- Query complexity: the number of oracle calls.
- Circuit size: the number of total quantum gates.

Deutsch–Josza problem

- There is a hidden Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ that is a **constant or balanced**.
- Quantum oracle $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$.
- Goal is to determine whether f is constant or balanced.
- Classical deterministic algorithm needs $2^{n-1} + 1$ oracle calls.
- Deutsch–Josza algorithm solves this problem by **single** oracle call (and $O(n)$ gates).

Deutsch–Josza algorithm



$$|0\rangle^{\otimes n} |1\rangle \xrightarrow{H^{\otimes(n+1)}} |+\rangle^{\otimes n} |-\rangle = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle |-\rangle$$

$$\xrightarrow{U_f} \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle |-\rangle$$

$$\xrightarrow{H^{\otimes(n+1)}} \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{z \in \{0,1\}^n} (-1)^{f(x)} (-1)^{\langle x, z \rangle} |z\rangle |1\rangle$$

The probability of outcome

$$\begin{aligned} & \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{z \in \{0,1\}^n} (-1)^{f(x)} (-1)^{\langle x, z \rangle} |z\rangle |1\rangle \\ &= \sum_{z \in \{0,1\}^n} \left(\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} (-1)^{\langle x, z \rangle} \right) |z\rangle |1\rangle \end{aligned}$$

The coefficient of $|0\rangle^{\otimes n} |1\rangle$ is $S := \frac{1}{2^n} \sum_x (-1)^{f(x)}$. Here, $S^2 = 1$ if $f(x)$ is constant, and $S^2 = 0$ if $f(x)$ is balanced.

In the Deutsch–Josza algorithm, the first n qubits are measured and output “constant” if all-zero is measured, and output “balanced”, otherwise.

Assignments (Deadline is Jan. 10)

- ① Describe quantum circuits computing the following Boolean functions, i.e., quantum circuit U satisfies

$$U |x\rangle |y\rangle |0\rangle^{\otimes w} = |x\rangle |y \oplus f(x)\rangle |0\rangle^{\otimes w}$$

for some w .

- Ⓐ $f(x_1, x_2, x_3, x_4) := x_1 \oplus x_2 \oplus x_3 \oplus x_4$.
 - Ⓑ $f(x_1, x_2, x_3, x_4) := x_1 \wedge x_2 \wedge x_3 \wedge x_4$.
 - Ⓒ $f(x_1, x_2, x_3, x_4) := (x_1 \vee x_2) \wedge (x_3 \vee x_4)$.
 - Ⓓ $f(x_1, x_2, x_3) := \text{Majority of } x_1, x_2 \text{ and } x_3$.
- ② (Advanced) For fixed $S \subseteq \{1, 2, \dots, n\}$, a Boolean function f is either $g_a(x) = a + \sum_{i \in S} x_i \pmod 2$ for $a \in \{0, 1\}$ or $h(x)$ satisfying $\sum_x (-1)^{g_0(x) + h(x)} = 0$. Show quantum algorithm that distinguishes the two cases $f(x) = g_0(x)$ or $g_1(x)$ and $f(x) = h(x)$ for $h(x)$ satisfying the above condition.