Almost All Quantum Oracles Are Impossible to Realize in Practice(V.2)

Robert R. Tucci
P.O. Box 226
Bedford, MA 01730
tucci@ar-tiste.com

February 15, 2009

Abstract

In this blog post, I give an introduction to quantum oracles from a quantum computer programmer's perspective. This being a blog post, I've tried to be as introductory and pedagogical as possible.

1 Introduction

In this blog post, I give an introduction to quantum oracles from a quantum computer programmer's perspective. This being a blog post, I've tried to be as introductory and pedagogical as possible. A quantum oracle is typically a unitary operator that takes $|0\rangle|x\rangle$ to $|f(x)\rangle|x\rangle$ for some function $f : \{0,1\}^{N_B} \to \{0,1\}$, where N_B is the number of bits. Many quantum algorithms call a quantum oracle one or more times.

Among other things, I will prove in this blog post that for almost all functions f, the corresponding quantum oracle has exponential(N_B) time complexity. Oracles are often compared to subroutines called by a larger computer program. If a computer program calls a subroutine that takes forever to run, the full computer program takes forever to run too, making it impossible to realize in practice. In the past, I've been often perplexed by papers that use quantum oracles without worrying about their time-complexity. This made me want to write a blog post about the subject of quantum oracles.

If a unitary operator is expressed as a **Sequence of Elementary Operations (SEO)** (by elementary operations we mean single-qubit rotations and CNOTs), then the number of CNOTs can be used as a measure of the time complexity of the operator. (Being two-body interactions, CNOTs take much longer to perform physically than single-qubit rotations, so we only count the former.) This is how we will measure the time-complexity of a unitary operator in this paper. If an operator can be expressed as a SEO with polynomial(N_B) many CNOTs, it will be said to have **polynomial complexity**. If exponential(N_B) many CNOTs are required to express it, it will be said to have **exponential complexity**.

2 Notation and Preliminaries

In this section, we will define some notation that is used throughout this paper.

We will often use the symbol N_B for the number (≥ 1) of qubits and $N_S = 2^{N_B}$ for the number of states with N_B qubits. The quantum computing literature often uses n for N_B and N for N_S, but we will avoid this notation. We prefer to use n for the number operator, defined below.

Let $Bool = \{0,1\}$. As usual, let $\mathbb{Z}, \mathbb{R}, \mathbb{C}$ represent the set of integers (negative and non-negative), real numbers, and complex numbers, respectively. For integers a, b such that $a \leq b$, let $Z_{a,b} = \{a, a+1, \ldots b-1, b\}$.

We will use $\Theta(S)$ to represent the "truth function"; $\Theta(S)$ equals 1 if statement S is true and 0 if S is false. For example, the Kronecker delta function is defined by $\delta_x^y = \delta(x,y) = \Theta(x = y)$.

Let $\overline{0} = 1$ and $\overline{1} = 0$. If $\vec{a} = a_{N_B-1} \ldots a_2 a_1 a_0$, where $a_\mu \in Bool$, then $dec(\vec{a}) = \sum_{\mu=0}^{N_B-1} 2^\mu a_\mu = a$. Conversely, $\vec{a} = bin(a)$.

We define the single-qubit states $|0\rangle$ and $|1\rangle$ by

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ , \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ . \tag{1}$$

If $\vec{a} \in Bool^{N_B}$, we define the N_B-qubit state $|\vec{a}\rangle$ as the following tensor product

$$|\vec{a}\rangle = |a_{N_B-1}\rangle \otimes \ldots |a_1\rangle \otimes |a_0\rangle \ . \tag{2}$$

For example,

$$|01\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \ . \tag{3}$$

I_k will represent the $k \times k$ unit matrix.

Suppose $\beta \in Z_{0,N_B-1}$ and M is any 2×2 matrix. We define $M(\beta)$ by

$$M(\beta) = I_2 \otimes \cdots \otimes I_2 \otimes M \otimes I_2 \otimes \cdots \otimes I_2 \ , \tag{4}$$

where the matrix M on the right hand side is located at qubit position β in the tensor product of N_B 2×2 matrices. The numbers that label qubit positions in the tensor product increase from right to left ($\leftarrow$), and the rightmost qubit is taken to be at position 0.

The Pauli matrices are

$$\sigma_X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \ , \ \sigma_Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \ , \ \sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \ . \tag{5}$$

The one-qubit Hadamard matrix H is defined as:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \ . \tag{6}$$

The number operator n for a single qubit is defined by

$$n = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1 - \sigma_Z}{2} \ . \tag{7}$$

Note that

$$n|0\rangle = 0|0\rangle = 0 \ \ , \ \ n|1\rangle = 1|1\rangle \ . \tag{8}$$

We will often use $\overline{n}$ as shorthand for

$$\overline{n} = 1 - n = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \frac{1 + \sigma_Z}{2} \ . \tag{9}$$

Define P_0 and P_1 by

$$P_0 = \overline{n} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = |0\rangle\langle 0| \ , \ \ P_1 = n = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = |1\rangle\langle 1| \ . \tag{10}$$

P_0 and P_1 are orthogonal projection operators and they add to one:

$$P_a P_b = \delta(a,b) P_b \quad \text{for } \ a, b \in Bool \ , \tag{11}$$

$$P_0 + P_1 = I_2 \ . \tag{12}$$

For $\vec{a} \in Bool^{N_B}$, let

$$P_{\vec{a}} = P_{a_{N_B-1}} \otimes \cdots \otimes P_{a_2} \otimes P_{a_1} \otimes P_{a_0} \ . \tag{13}$$

For example, with 2 qubits we have

$$P_{00} = P_0 \otimes P_0 = diag(1,0,0,0) \ , \tag{14}$$

$$P_{01} = P_0 \otimes P_1 = diag(0,1,0,0) \ , \tag{15}$$

$$P_{10} = P_1 \otimes P_0 = diag(0,0,1,0) \ , \tag{16}$$

$$P_{11} = P_1 \otimes P_1 = diag(0,0,0,1) \ . \tag{17}$$

Note that

$$P_{\vec{a}} P_{\vec{b}} = \delta(\vec{a}, \vec{b}) P_{\vec{b}} \quad \text{for } \ \vec{a}, \vec{b} \in Bool^{N_B} \ , \tag{18}$$

$$\sum_{\vec{a} \in Bool^{N_B}} P_{\vec{a}} = I_2 \otimes I_2 \otimes \cdots \otimes I_2 = I_{2^{N_B}} \ . \tag{19}$$

Let $P_{\cdot} = \sum_{a \in Bool} P_a = I_2$. More generally, for $\vec{b} \in Bool^{N_B}$, if one or more of the binary indices of $P_{\vec{b}}$ is replaced by a dot, this will signify that those indices are summed over. For example, $P_{0\cdot 1\cdot} = \sum_{a,b \in Bool} P_{0a1b}$. Henceforth, we will denote the set $\{0, 1, \cdot\}$ by $BoolDot$ and use Latin letters for elements of $Bool$ and Greek letters for elements of $BoolDot$. Note that $P_0 P_{\cdot} = P_0$ and $P_{\cdot} P_{\cdot} = P_{\cdot}$. More generally, if $\mu, \nu \in BoolDot$, then

$$P_\mu P_\nu = P_\mu \delta_\mu^\nu + (1 - \delta_\mu^\nu)(\delta_\mu^{\cdot} P_\nu + \delta_\nu^{\cdot} P_\mu) \ . \tag{20}$$

A special case of the last equation is

$$P_a P_\nu = (\delta_a^\nu + \delta_{\cdot}^\nu) P_a \tag{21}$$

for $a \in Bool$ and $\nu \in BoolDot$.

Next we explain our circuit diagram notation. We label single qubits (or qubit positions) by a Greek letter or by an integer. When we use integers, the topmost qubit wire is 0, the next one down is 1, then 2, etc. *Note that in our circuit diagrams, time flows from the right to the left of the diagram.* Careful: Many workers in Quantum Computing draw their diagrams so that time flows from left to right. We eschew their convention because it forces one to reverse the order of the operators every time one wishes to convert between a circuit diagram and its algebraic equivalent in Dirac notation.

Suppose $U \in U(2)$. If τ and κ are two different qubit positions, gate $U(\tau)^{n(\kappa)}$ (or $U(\tau)^{\overline{n}(\kappa)}$) is called a **controlled** U with target τ and control κ. When $U = \sigma_X$, this reduces to a **CNOT (controlled NOT)**. If τ,κ_1 and κ_0 are 3 different qubit positions, $\sigma_X(\tau)^{n(\kappa_1)n(\kappa_0)}$ is called a **Toffoli gate** with target τ and controls κ_1, κ_0. Suppose $N_K \geq 2$ is an integer and $\vec{b} \in Bool^{N_K}$. Suppose $\tau, \kappa_{N_K-1}, \kappa_{N_K-2}, \ldots, \kappa_1, \kappa_0$ are distinct qubits and $\vec{\kappa} = (\kappa_{N_K-1}, \kappa_{N_K-2}, \ldots, \kappa_1, \kappa_0)$. Gate $U(\tau)^{P_{\vec{b}}(\vec{\kappa})}$ is called a **multiply controlled** U with target τ and N_K controls $\vec{\kappa}$. When $U = \sigma_X$, this reduces to an **MCNOT (multiply controlled NOT)**.

3 Standard Oracles

In this section, we define and discuss standard quantum oracles.

Consider a function $f : Bool^{N_B} \to Bool$. Let $\tau, \beta_0, \beta_1, \ldots, \beta_{N_\beta-1}$ denote N_B+1 distinct qubit positions. Let $\vec{b} = (b_{N_\beta-1}, \ldots, b_1, b_0) \in Bool^{N_B}$, $\vec{\beta} = (\beta_{N_\beta-1}, \ldots, \beta_1, \beta_0)$, and $P_{\vec{b}}(\vec{\beta}) = \prod_{j=0}^{N_\beta-1} P_{b_j}(\beta_j)$. We define a **standard quantum oracle** Ω with target qubit τ and N_B control qubits $\vec{\beta}$ by

$$\begin{aligned} \Omega &= \sigma_X(\tau)^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})} && \text{(22a)} \\ &= \prod_{\vec{b}} \sigma_X(\tau)^{f(\vec{b}) P_{\vec{b}}(\vec{\beta})} && \text{(22b)} \\ &= \sum_{\vec{b}} P_{\vec{b}}(\vec{\beta}) \;\; \sigma_X(\tau)^{f(\vec{b})} \; . && \text{(22c)} \end{aligned}$$

We've expressed Ω in 3 equivalent forms, the exponential, product and sum forms. The equivalence of these forms is readily established by applying $|\vec{b}\rangle_{\vec{\beta}}$ to the right hand side of each form. Note that we can "pull" the $\vec{b}$ sum out of the exponential, but only if we also pull out the projector $P_{\vec{b}}$.

Note that for $t \in Bool$ and $\vec{b} \in Bool^{N_B}$,

$$\Omega |t\rangle_\tau |\vec{b}\rangle_{\vec{\beta}} = \sigma_X(\tau)^{f(\vec{b})} |t\rangle_\tau |\vec{b}\rangle_{\vec{\beta}} = |t \oplus f(\vec{b})\rangle_\tau |\vec{b}\rangle_{\vec{\beta}} \;\; , \tag{23}$$

where $\oplus$ denotes mod 2 summation. The property Eq.(23) of Ω is often utilized in

quantum algorithms, when Ω is used to "load input data into a register": $\Omega|0\rangle_\tau|\vec{b}\rangle_{\vec{\beta}} = |f(\vec{b})\rangle_\tau|\vec{b}\rangle_{\vec{\beta}}$. From Eq.(23), it follows that the matrix elements of Ω are

$$\Omega(t',\vec{b}'|t,\vec{b}) = \langle t'|_\tau\langle\vec{b}'|_{\vec{\beta}}\ \ \Omega\ \ |t\rangle_\tau|\vec{b}\rangle_{\vec{\beta}} = \delta^{t'}_{t\oplus f(\vec{b})}\ \delta^{\vec{b}'}_{\vec{b}}\ . \tag{24}$$

Thus, Ω can be represented as the following matrix

$$\Omega = \begin{array}{c|c|c} & t=0 \\ & \vec{b} & t=1 \\ & & \vec{b} \\ \hline t'=0 & & \\ \vec{b}' & \overline{\Delta} & \Delta \\ \hline t'=1 & & \\ \vec{b}' & \Delta & \overline{\Delta} \end{array}\ , \tag{25}$$

where Δ is a diagonal matrix whose diagonal entries are either 0 or 1 and where $\overline{\Delta} = 1 - \Delta$.

Another common quantum oracle is Ω' defined by

$$\Omega' = (-1)^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})}\ . \tag{26}$$

The oracle in the original Grover's algorithm is simply Ω' with $f(\vec{b}) = \delta(\vec{b},\vec{b}_0)$, where $\vec{b}_0$ is the so called target state. Ω' can be easily expressed in terms of Ω by introducing an ancilla target qubit τ:

$$\begin{aligned} \Omega'(\vec{\beta})|0\rangle_\tau &= \sigma_X(\tau)\sigma_Z(\tau)^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})}\sigma_X(\tau)|0\rangle_\tau & \text{(27a)} \\ &= \sigma_X(\tau)H(\tau)\left[\sigma_X(\tau)^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})}\right] H(\tau)\sigma_X(\tau)|0\rangle_\tau\ . & \text{(27b)} \end{aligned}$$

Another common quantum oracle is Ω'' defined by

$$\begin{aligned} \Omega'' &= [e^{ig\sigma_X(\alpha)}]^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})} & \text{(28a)} \\ &= \quad \vec{\beta} \ \text{—} \boxed{\textstyle\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}} \text{—} \\ &\quad\ \ \alpha \ \text{—} \boxed{e^{ig\sigma_X}} \text{—} \end{aligned}\ , \tag{28b}$$

where $g \in \mathbb{R}$ and $\alpha, \vec{\beta} = (\beta_{N_B-1}, \ldots, \beta_1, \beta_0)$ are distinct qubits. Ω'' can be easily expressed in terms of two Ω's by introducing an ancilla target qubit τ:

$$\Omega''(\vec{\beta},\alpha)|0\rangle_\tau = \left[\sigma_X(\tau)^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})}\right] e^{ig\sigma_X(\alpha)n(\tau)} \left[\sigma_X(\tau)^{\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})}\right] |0\rangle_\tau \tag{29a}$$

$$= \begin{array}{ll} & \vec{\beta} \\ & \tau \\ & \alpha \end{array} \begin{array}{c} \text{—} \boxed{\textstyle\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}} \text{—} \boxed{\textstyle\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}} \text{—} \\ \text{—} \times \text{—}\bullet\text{—} \times \text{—} \\ \text{—} \boxed{e^{ig\sigma_X}} \text{—} \end{array} |0\rangle_\tau \tag{29b}$$

Note that the initial α and $\vec{\beta}$ qubit "worldlines" carry "exotic" operators like $e^{ig\sigma_X(\alpha)}$ and $\sum_{\vec{b}} f(\vec{b})P_{\vec{b}}(\vec{\beta})$, whereas the auxiliary τ worldline carries only "non-exotic" operators like $\sigma_X(\tau)$ and $n(\tau)$. The reason we need two Ω's to represent Ω'' is that the second Ω reverses the change made by the first to the auxiliary τ qubit.

4 Inherited Oracles

In this section we introduce a class of quantum oracles inherited from classical reversible computing.

For any function $f : Bool^{N_B} \rightarrow Bool$, we can define a matrix $T(t|\vec{b}) = \delta^t_{f(\vec{b})}$ where $t \in Bool$ and $\vec{b} \in Bool^{N_B}$. In classical reversible computing, the matrix $T(t|\vec{b})$ is extended to an operator $\mathcal{T}(t, \vec{b}_-|t_-, \vec{b}) \in Bool$ that is reversible ($\mathcal{T}\mathcal{T}^T = 1$) and that satisfies

$$\sum_{\vec{b}_-} \mathcal{T}(t, \vec{b}_-|t_- = 0, \vec{b}) = \delta^t_{f(\vec{b})} \ , \tag{30}$$

where $\vec{b}, \vec{b}_- \in Bool^{N_B}$ and $t, t_- \in Bool$. (Think of $T(t|\vec{b})$ and $\mathcal{T}(t, \vec{b}_-|t_-, \vec{b})$ as conditional probabilities.) t_- is called a source index and $\vec{b}_-$ a sink index. Collectively, t_- and $\vec{b}_-$ are called ancilla indices.

Now note that by virtue of Eq.(24), Eq.(30) is satisfied if we set

$$\mathcal{T}(t, \vec{b}_-|t_-, \vec{b}) = |\Omega(t, \vec{b}_-|t_-, \vec{b})|^2 = \Omega(t, \vec{b}_-|t_-, \vec{b}) \ . \tag{31}$$

(Think of $\Omega(t, \vec{b}_-|t_-, \vec{b})$ as a probability amplitude). There are many possible ways of extending T to $\mathcal{T}$, and the standard oracle is one of them. In general, any circuit used in classical reversible computing can be transformed to the standard oracle canonical form, and used immediately as a quantum oracle component of a quantum circuit. Thus, for any $f : Bool^{N_B} \rightarrow Bool$, if an algorithm of polynomial complexity for calculating $f(\vec{b})$ is known, then we can always construct from this algorithm a quantum oracle of polynomial complexity. For example, if $f(\vec{b})$ is a polynomial function, it has an exact quantum oracle of polynomial complexity. If $f(\vec{b})$ is well approximated by a polynomial function, then it has an approximate quantum oracle of polynomial complexity.

5 Banded Oracles

In this section we introduce a special class of quantum oracles that I like to call "banded oracles".[1] Such oracles can be expressed as a SEO with polynomial(N_B)

[1]Banded oracles have been implemented in the Java application QuanOracle, written by the author of this paper. Ref.[1] describes how to operate QuanOracle. QuanOracle is a member of

many CNOTs.

For $\vec{b} \in Bool^{N_B}$, if $j = dec(\vec{b})$, let $P_j = P_{\vec{b}}$. For some integer $j \geq 0$, define a single band of P's, from 0 to j, by

$$P_{[0,j]} = \sum_{k=0}^{j} P_k \ . \tag{32}$$

We can use the identity $P_0 + P_1 = P_{\cdot}$ to reduce the number of projection operators on the right hand side of Eq.(32). For example, for $10 = dec(1010)$ and $11 = dec(1011)$,

$$P_{[0,1010]} = P_{0\cdots} + P_{100\cdot} + P_{1010} \ , \tag{33}$$

and

$$P_{[0,1011]} = P_{0\cdots} + P_{100\cdot} + P_{101\cdot} \ . \tag{34}$$

The right hand sides of Eqs.(33) and (34) are readily apparent if you think of the way one counts in binary.

Next, for some integers j_1, j_2 such that $0 \leq j_1 \leq j_2$, define a single band of P's, from j_1 to j_2, by

$$P_{[j_1,j_2]} = P_{[0,j_1-1]} + P_{[0,j_2]} \ . \tag{35}$$

Addition on the right hand side of Eq.(35) is taken to be mod 2 so $P_0 + P_0 = P_1 + P_1 = 0$ can be used to cancel overlapping P's. For example, for $11 = dec(1011)$ and $14 = dec(1110)$,

$$P_{[1011,1110]} = (P_{0\cdots} + P_{100\cdot} + P_{1110}) + (P_{0\cdots} + P_{10\cdot\cdot} + P_{110\cdot} + P_{1110}) \ . \tag{36}$$

In the right hand side of Eq.(36), the two $P_{0\cdots}$ cancel.

The above definition of a single band of P's can be generalized in the obvious way to define multiple non-overlapping bands of P's.

When the exponent $\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})$ of Ω equals a sum of multiple non-overlapping bands of P's, then we get what I call a banded oracle. The number of bands must be independent of N_B (or be $\leq$ polynomial(N_B)). Clearly, a banded oracle can be expressed as a product of m MCNOTs, where m is $\leq$ polynomial(N_B). Each of these MCNOTs has N_B or fewer controls, so it can be expressed as a SEO with polynomial(N_B) many CNOTs. Thus, the oracle itself can be expressed as a SEO with polynomial(N_B) many CNOTs.

---

a suite of Java applications called QuanSuite. The QuanSuite applications and source code are available for free at www.ar-tiste.com.

6 Impossible Oracles

In the previous section we stipulated that the exponent $\sum_{\vec{b}} f(\vec{b}) P_{\vec{b}}(\vec{\beta})$ for banded oracles be expressible as sum of m P's, where m is $\leq$ polynomial(N_B). One wonders if this is also the case for any standard oracle. The answer is a resounding no. As we show next, the vast majority of standard oracles require exponential(N_B) many P's in their exponent.

Let $\vec{b} \in Bool^{N_B}$ and $f_{\vec{b}} = f(\vec{b}) \in Bool$. Suppose N is a positive integer. For each $k \in Z_{0,N}$, let $c^{(k)} \in Bool$ and $\vec{\alpha}^{(k)} = (\alpha^{(k)}_{N_B-1}, \ldots, \alpha^{(k)}_1, \alpha^{(k)}_0) \in BoolDot^{N_B}$. Suppose we have found an N, and for each $k \in Z_{1,N}$, a pair $(c^{(k)}, \vec{\alpha}^{(k)})$, such that

$$\sum_{\vec{b}} f_{\vec{b}} P_{\vec{b}} = \sum_{k=1}^{N} c^{(k)} P_{\vec{\alpha}^{(k)}} \ . \tag{37}$$

Applying Eq.(21) to Eq.(37) yields

$$f_{\vec{b}} = \sum_{k=1}^{N} c^{(k)} \prod_{j=0}^{N_B-1} \{\delta_{b_j}^{\alpha_j^{(k)}} + \delta_{\cdot}^{\alpha_j^{(k)}}\} \ \text{ for each } \vec{b} \in Bool^{N_B}. \tag{38}$$

Eqs.(38) define a function Φ such that

$$\Phi(c^{(k)}, \vec{\alpha}^{(k)})_{\forall k \in Z_{1,N}} = (f_{\vec{b}})_{\forall \vec{b} \in Bool^{N_B}} \ . \tag{39}$$

For example, for $N_B = 3$ and $N = 2$, one has

$$\sum_{\vec{b} \in Bool^3} f_{\vec{b}} P_{\vec{b}} = c^{(1)} P_{\alpha_2^{(1)} \alpha_1^{(1)} \alpha_0^{(1)}} + c^{(2)} P_{\alpha_2^{(2)} \alpha_1^{(2)} \alpha_0^{(2)}} \ , \tag{40}$$

$$\begin{cases} f_{000} = \sum_{k=1}^{2} c^{(k)} (\delta_0^{\alpha_2^{(k)}} + \delta_{\cdot}^{\alpha_2^{(k)}})(\delta_0^{\alpha_1^{(k)}} + \delta_{\cdot}^{\alpha_1^{(k)}})(\delta_0^{\alpha_0^{(k)}} + \delta_{\cdot}^{\alpha_0^{(k)}}) \\ f_{001} = \sum_{k=1}^{2} c^{(k)} (\delta_0^{\alpha_2^{(k)}} + \delta_{\cdot}^{\alpha_2^{(k)}})(\delta_0^{\alpha_1^{(k)}} + \delta_{\cdot}^{\alpha_1^{(k)}})(\delta_1^{\alpha_0^{(k)}} + \delta_{\cdot}^{\alpha_0^{(k)}}) \\ \vdots \\ f_{111} = \sum_{k=1}^{2} c^{(k)} (\delta_1^{\alpha_2^{(k)}} + \delta_{\cdot}^{\alpha_2^{(k)}})(\delta_1^{\alpha_1^{(k)}} + \delta_{\cdot}^{\alpha_1^{(k)}})(\delta_1^{\alpha_0^{(k)}} + \delta_{\cdot}^{\alpha_0^{(k)}}) \end{cases} \tag{41}$$

and

$$\Phi(c^{(k)}, \alpha_2^{(k)}, \alpha_1^{(k)}, \alpha_0^{(k)})_{\forall k \in Z_{1,2}} = (f_{000}, f_{001}, \ldots, f_{111}) \ . \tag{42}$$

In general,

$$\Phi : (Bool \times Bool^{N_B})^N \to Bool^{2^{N_B}} \ , \tag{43}$$

where Φ is not necessarily a surjection (onto). The range of Φ can be identified with the set of all possible functions $f : Bool^{N_B} \to Bool$. The size of Φ's domain is $2^{(N_B+1)N}$, whereas the size of its range is $2^{2^{N_B}}$. For large N_B, if N is polynomial(N_B),

then the image of Φ only covers a vanishingly small fraction of the range of Φ. Thus, almost all quantum oracles require exponential(N_B) many P's in their exponent. This means that for large N_B, almost all quantum oracles are impossible to realize in practice.

7 Acknowledgements

In updating from version 1 to version 2, I profitted greatly from blog comments by Scott Aaronson.

References

[1] R.R.Tucci, "QuanFou, QuanGlue, QuanOracle and QuanShi, Four Special Purpose Quantum Compilers", arXiv:0802.2367