Concretely Efficient Reductions From Quantum Circuits

(preliminary version, in preparation and awaiting experimental results)

Chaowen Guan and Kenneth W. Regan

Department of CSE, University at Buffalo, Amherst, NY 14260 USA
{chaoweng,regan}@buffalo.edu

August 28, 2017

Abstract

We give new explicit conversions from a quantum circuit C into a small set of Boolean formulas ϕ_k such that the acceptance amplitude of C (on a given input) can be computed from the numbers of satisfying assignments to the ϕ_k. They enable use of heuristic #SAT solvers to perform emulation of quantum circuits. We likewise represent the acceptance probability as a difference $\#sat(\phi_1) - \#sat(\phi_2)$ in a way that facilitates sampling of measured values $C(x)$. Computational experiments are performed and compared with other quantum circuit emulators on a variety of instances, including formulas arising from stages of executing the quantum inner loop of Shor's algorithm, which is otherwise classical.

1 Introduction

Quantum circuits can efficiently perform n-bit computations that are commonly believed to require time exponential in n on classical computers, in particular those associated to the problem of factoring n-bit integers. They are not however known or believed to solve NP-hard problems with comparable efficiency. The theoretical concepts involved are asymptotic, talking about growth rates as n gets large, but the concrete time for moderate values of n makes itself felt. Current standards for cryptographic use of factoring involve n on the order of $1,000$. Success in classical emulation in many cases has recently been claimed for n approaching 50 [HSST, BIS+16]. It is thus important to develop and test emulations that can work for problem sizes in this range of n.

Simulators and emulators use several forms. One is to go straight to the $N \times N$ matrices defining compositions of gate operations, where $N = 2^n$, but use sparse representations of these matrices. Others first pre-process a given circuit C in time scaling as $n^{O(1)}$ before going to steps that scale as N. We treat and develop emulators that convert C into a completely different object to which off-the-shelf methods can be applied. In [DHH+04, GS06, RC12] the object is a small set of polynomial equations $p_C = 0$. In this paper we produce small sets of Boolean formulas $\phi_k^C(z_1, \ldots, z_r)$ (in conjunctive normal form) such that C can be simulated with exact knowledge of the number of assignments in $\{0,1\}^r$ that satisfy ϕ_j^C.

The tasks of counting the number of solutions to a given polynomial equation and the number of satisfying assignments to a Boolean formula belong to a complexity class of functions called

#P. The corresponding class of decision problems is called PP. The general cases of these tasks are both NP-hard. Although NP-hardness has generally been regarded as strong evidence of *asymptotic* intractability, recently there have been broad advances on solving *concrete* cases of these tasks. Most of this success has come from so-called SAT-solvers asked to find just one satisfying assignment, but recently so-called #SAT solvers charged with counting the number of satisfying assignments *exactly* have gained traction.

Quantum circuits yield special cases of these tasks that are captured by a subclass of PP called AWPP [FR98] which unlike PP is not known or believed to contain NP-hard problems. However, AWPP is defined "semantically" via a so-called "promise condition" (see [FFKL03]) and there does not seem to be a simple *syntactic* way to distinguish the objects we will produce from general ones given to #SAT solvers. Our prime interest is nevertheless in whether these instances are concretely easier than general ones belonging to common NP-hard tasks. Most in particular, direct attempts to apply SAT-solvers on classical NP-based representations of the factoring problem have not led to success (cf. [Ask14]). Direct encoding of factoring via Karatsuba multiplication is called a "tough case" by [YB17]. Our formulas ϕ_j^C will be substantially different in character, and will be used to drive the quantum inner loop of Shor's algorithm [Sho94], the rest of which is classical, rather than encode the factoring predicate directly. Thus our goal will be to test whether the resulting classical attack on factoring is any more dangerous than others that have been tried.

Of course the above facts imply that there *exists* a reduction from C to formulas ϕ_j^C. One could be obtained by applying the generic Cook-Levin reduction to $\#SAT$, starting from the polynomials p_C for instance. However, we seek the most efficient reductions, ones that are *natural*, *specific*, and *tight*. We will use the technique that produces p_C as a proof guide. Some theoretical interest may come from structural properties of the particular ϕ_j^C that we obtain, including a certain sameness of form in the Hadamard and Toffoli gate cases. We presume familiarity with quantum circuits and gates; references include [BBC+95, NC00, LR14]. We will consider three prominent universal quantum gate sets:

- Hadamard (H) and Toffoli (Tof) gates.
- Hadamard and controlled quarter-circle phase (CS) gates.
- Hadamard, controlled-NOT (CNOT), and T-gates (T).

It is commonplace to include CNOT in the first two sets as well. The first group involve phases 0 and π only, represented by entries $+1$ and -1 (divided by $\sqrt{2}$ in the case of the Hadamard gate) in the corresponding matrices. The second group adds phases $\pi/2$ and $-\pi/2$ represented by entries i and $-i$. The third group needs minimum phase angle $\pi/4$, which will correspond to $k = 3$ and $K = 2^k = 8$ in notation to come, but is more expressive.

Two important technical points concern approximation and sampling. The expression for the acceptance amplitude in the first group involves formulas ϕ_0, ϕ_1 with some number r of variables—h of them "free"—and gives acceptance amplitudes of the form

$$\frac{\#sat(\phi_0) - \#sat(\phi_1)}{R},$$

where $R = 2^{h/2}$ rather than be of order 2^h or 2^r. The total numbers n_0 and n_1 of satisfying assignments to ϕ_0 and ϕ_n will each have order 2^h, but their difference is *a priori* constrained to

be at most R. It will therefore not suffice to compute n_0 to within a factor of $(1+\epsilon)$, say, nor likewise n_1. This is why exact #SAT solving is sought.

On the other hand, many prominent quantum algorithms—Shor's among them—need quantum only to generate samples z from a distribution $\mathcal{D}$ on $\{0,1\}^n$. Here approximations z' to z are often tolerated. Hence we are most narrowly interested in the *sampling* problem for satisfying assignments, indeed cases of *uniform generation*. Exact counting generally implies uniform sampling, but when and whether the latter affords more slack is a difficult problem in general. The second point is that to imitate the classical reduction from uniform sampling to $\#sat$ we need parallel results that give the acceptance probability, rather than the amplitude, as a difference

$$\frac{\#sat(\psi_0) - \#sat(\psi_1)}{R^2},$$

where ψ_0 and ψ_1 are "double-rail" versions of ϕ_0 and ϕ_1. Whether further savings can be realized by further use of approximation leads to further questions both about sampling and the workings of individual heuristic #SAT solvers

[This draft lays out the proofs; experiments to come...]

2 Overview and General Theorem

Let C be a quantum circuit on m qubits with s gates and put $M = 2^m$. Consider the formal product of the s-many unitary $M \times M$ matrices U_ℓ, one for each gate in C. It expands to a sum of s-fold products of matrix entries. Every nonzero product can be called a *possible path* though the object described by C. The value of the product is a complex number $re^{i\theta}$ with *phase* θ. One thing to note is that it is customary to write the input a to C on the left and list the gates/gate-matrices left-to-right as $U_1, \ldots, U_s$, but the matrix computations are

$$C(a) = U_s U_{s-1} \cdots U_2 U_1 a \qquad \text{and} \qquad \langle\, a \,|\, C \,|\, b \,\rangle = \langle C(a), b \rangle.$$

If a path begins in row i of a, then it enters U_1 through column i and exits through some row j, whereupon it enters U_2 through column j. For intuition we might wish to use either the transposed computation or the conjugate transpose,

$$C(a) = a^T U_1^T U_2^T \cdots U_{s-1}^T U_s^T \qquad \text{or} \qquad C(a)^* = a^* U_1^* U_2^* \cdots U_{s-1}^* U_s^*,$$

and talk about the path entering row i of the first matrix and exiting via column j, etc., so as to align with how we read the circuit. However, we pay strict accord to how the computation unfolds.

We make two mild assumptions about U_ℓ and θ. The first, called *balance*, is that all nonzero entries of U_ℓ have the same magnitude. This property is preserved under tensor products, so it suffices to verify it for the $2^r \times 2^r$ matrix defining an r-ary gate locally. The second is that θ be an integer multiple of $2\pi/K$ where $K = 2^k$ for some k. Then $e^{2\pi i/K}$ is a primitive K-th root of unity, which we denote by ω when k is understood. When k is least possible we call either $2\pi/K$ or $1/K$ the *min-phase* of the circuit and identify the phases with $0, \ldots, K-1$ modulo K.

If $x = x_1, \ldots, x_n$ are variables in a Boolean formula ϕ and $a \in \{0,1\}$, then $\phi[x = a]$ stands for the formula obtained by substituting a_i for x_i for each i and simplifying operations involving constants. We *could* make a distinction between $\#sat(\phi[x = a])$ referring to assignments in the reduced formula—not including $x_1, \ldots, x_n$—and $\#sat(\phi)[x = a]$ meaning the number of

assignments to the original ϕ that give the values $a_1, \ldots, a_n$ to $x_1, \ldots, x_n$, respectively. However, the following base-case understanding removes the need: A formula ϕ with zero variables has $2^0 = 1$ assignment λ belonging to $\{0,1\}^0 = \{\lambda\}$. If ϕ is equivalent to $\top$, the constant true formula, then λ is a satisfying assignment; if $\phi \equiv \bot$, the constant false formula, then λ is unsatisfying. An empty conjunction defaults to $\top$ while an empty disjunction or XOR defaults to $\bot$. Now we can state and already prove the main general theorem:

Theorem 1. *Let C be a circuit of m qubits and s balanced gates of minphase $1/K = 2^{-k}$ and maximum arity $r \leq m$ qubits. Then we can efficiently build a Boolean formula ϕ of size $O(m + sk2^{2r})$ in variables $\vec{w}, \vec{x}, \vec{y}, \vec{z}$ and find a constant R such that for all $a, b \in \{0,1\}^m$:*

$$\begin{aligned} \langle a \,|\, C \,|\, b \rangle &= \frac{1}{R} \sum_{L=0}^{K-1} \#sat(\phi[\vec{w} = L, \vec{x} = a, \vec{z} = b])\omega^L \\ &= \frac{1}{R} \sum_{L,c} \phi_C(\vec{w} = L, \vec{x} = a, \vec{y} = c, \vec{z} = b])\omega^L. \end{aligned}$$

Moreover, for every assignment (a, c) to $\vec{x}, \vec{y}$ there is exactly one pair (L, b) such that $\phi_C[\vec{w} = L, \vec{x} = a, \vec{y} = c, \vec{z} = b]$ holds and it can be found in $O(s)$ time.

Proof. We track paths in stages $\ell = 1$ to s as they begin in a column $a = J_0 \in \{0,1\}^m$ of U_1 and terminate in row $b = I_s$ of U_s. We allocate $s + 1$ suites $W_0, \ldots, W_s$ of variables $w_{0,\ell}, \ldots, w_{k-1,\ell}$ which collectively track the phase $L \in \{0, \ldots, K-1\}$ of a path by $L = \sum_{j=0}^{k-1} w_j 2^j$. At each stage ℓ we identify m *location* literals $u_1, \ldots, u_m$ on the qubit lines whose values determine an entry column $J \in \{0,1\}^m$ to the matrix U_ℓ; initially they are the input variables $x_1, \ldots, x_m$. We allocate up to m fresh variables $y_1, \ldots, y_m$ whose values $I \in \{0,1\}^m$ stand for possible exit rows $I\ell$, which become either the entry column $J_{\ell+1}$ for the next stage or are equated with the output variables $z_1, \ldots, z_m$. (Note that the "u_i" will be meta-symbols, and extended constructions will allow them to be negated variables. We will also later distinguish between allocated variables y_i whose values are forced not free, calling them v_i instead.)

If J is any column value in $\{0,1\}^m$, then u_J denotes the unique conjunction of signed literals $\pm u_i$ (over $i = 1$ to m) whose value is 1 on J and 0 for all $J' \neq J$. For instance, if $J = 01101$ then $u_J = (\bar{u}_1 \wedge u_2 \wedge u_3 \wedge \bar{u}_4 \wedge u_5)$. We denote row conjuncts v_I similarly. Entering stage ℓ of the circuit, we consider all possible current phases $p_{\ell-1}$ coded by the variables $W_{\ell-1} = w_{0,\ell}, \ldots, w_{k-1,\ell}$. For all pairs I, J we add clauses as follows:

- If $U_\ell[I, J] = 0$ then we add $\neg(u_J \wedge v_I)$, which becomes a clause of $2m$ disjoined literals.
- If $U_\ell[I, J] = re^{i\theta}$, then by balance, $r \neq 0$ is independent of I, J and $\theta = 2\pi id/K$ for some $d \in [K]$. Then we add for $j = 0$ to $k - 1$ the clause

$$((u_J \wedge v_I) \rightarrow (w_{j,\ell} = w_{j,\ell-1} \oplus F_d(W_{\ell-1}))),$$

  where F_d is the fixed finite function true on all c such that $c + d$ causes a flip in bit j.

Note that F_d can be a function of the variables $w_{0,\ell-1}, \ldots, w_{j,\ell-1}$ alone. We can alternately consider that over $j = 0$ to $k - 1$ alone we have added the single clauses

$$w_{j,\ell} = w_{j,\ell-1} \oplus F'(u_1, \ldots, u_m, v_1, \ldots, v_m, w_0, \ldots, w_j),$$

where F' takes into account all the phases d that arise in the matrix entries $U_\ell[I, J]$ as specified by the value J for $u_1, \ldots, u_m$ and I for $v_1, \ldots, v_m$. Economizing F' will occupy much of the remainder of the paper, but for this proof we reason about F_d for all the u_J and v_I.

Finally we note that $v_1, \ldots, v_m$ become "$u_1, \ldots, u_m$" for the next stage if there is one, else we conjoin the clauses $\wedge_{i=1}^m (v_i = z_i)$ (or just substitute $z_1, \ldots, z_m$ directly). The last act is to add the clauses $\wedge_j \bar{w}_{j,0}$ and declare $\vec{w}$ in the theorem statement to refer to the terminal $w_{j,s}$ phase variables. Then $\vec{y}$ in the theorem statement ranges over $w_{j,\ell}$ for $1 \leq \ell \leq s-1$ and variables $v_{i,\ell}$ introduced as "v_i" in the corresponding stages ℓ. (We will pin it down further in specific instances later.) This finishes the construction of ϕ_C.

To see that it is correct, first consider any path P from a to b whose phase changes by L. First we substitute $\vec{x} = a$ and $\vec{z} = b$ and $W_s = L$. In the base case $s = 0$ with empty circuit, P can only be a path from a to $b = a$ with $L = 0$. Then we have $W_s = W_0$ and substituting L gives $\top$ if $b = a$ and $L = 0$, $\bot$ otherwise. For $s \geq 1$, to P there corresponds a unique assignment of row and column values

$$a = J_1, \quad I_1 = J_2, \quad \ldots, \quad I_{s-1} = J_s, \quad I_s = b$$

to literals designated "u_i" and "v_i" at each stage ℓ. For all $(I, J) \neq (I_\ell, J_\ell)$, all clauses $(u_I \wedge v_J) \rightarrow \ldots)$ are vacuously satisfied. This leaves the clause

$$((u_{J_\ell} \wedge v_{I_\ell}) \rightarrow (w_{j,\ell} = w_{j,\ell-1} \oplus F_d(W_{\ell-1}))),$$

where d is the phase of the nonzero entry $U_\ell[I, J]$. By induction, the values of $W_{\ell-1}$ in the assignment either have the phase c of the path entering that stage or the assignment is already determined to be unsatisfying. These determine the value $F_d(W_{\ell-1}$ and hence collectively over j these clauses determine that W_ℓ must have the correct value $c + d$ modulo K, else they are not satisfied. Since the values of the variables in W_ℓ are forced, we have a unique continuation of a satisfying assignment. In the last stage, the current phase value must become L. Hence we have mapped P to one satisfying assignment of $\phi_C[\vec{x} = a, \vec{z} = b, \vec{w} = L]$ (with W_0 already substituted to zeros).

Going the other way, suppose y is any satisfying assignment to ϕ_C (again with $W_0 = 0$). We argue that y maps uniquely to a path P_y. We get $a = J_1$ from the values assigned to $\vec{x}$, then the values $J_2, \ldots, J_s$ of the other column entries, and finally the exit row I_s which gives a b. The values of phases along the path are likewise determined by the assignment and must be correct. Hence the assignment yields a unique path. The path must be legal: at any stage the left-hand side of one clause of the form $(u_J \wedge v_I) \rightarrow \ldots$ holds so its consequent must be made true.

Thus the correspondence of counting paths and counting satisfying assignments is parsimonious for each phase value L, so the equation in Theorem 1 follows. Finally, we may observe that if U_ℓ is a tensor product of a $2^r \times 2^r$ matrix and identity matrices, then whenever I and I' vis-à-vis J and J' agree on the r qubit lines touched by the gate, their clauses can be identified, leaving at most 2^{2r} distinct clauses added at stage ℓ. The rest of the size estimation is straightforward. □

As already remarked, the main purpose of this paper is to find the most economical (and elegant) condensations for specific families of quantum gates. We also note that any initialization L_0 can be used for W_0 provided the corresponding target for W_s is shifted to be $L + L_0$. Here we finish by noting one further general feature of the emulation that already follows from this proof.

For any set B of target output values and phase L, we can define $p_C^L(a,B) = \sum_{b\in B} p_C^L(a,b)$, where $p_C^L(a,b)$ denotes the number of paths from a to b having phase L. We will find it convenient to maintain these sets of paths when B is a *cylinder*, that is, for some $I \subseteq \{1,\ldots,m\}$, $I = \{i_1,\ldots,i_r\}$, and binary string c of length r:

$$B = \{\, b \in \{0,1\}^m : b_{i_1} = c_1 \wedge b_{i_2} = c_2 \wedge \cdots \wedge b_{i_r} = c_r \,\}.$$

Singleton sets $\{b\}$ have this form with $I = \{1,\ldots,m\}$ and $c = b$, as do sets $B_i = \{\, b : b_i = 1 \,\}$ which represent measuring the single qubit i to test for a 1 value. Cylinders are important because we can choose *not* to substitute all $z_1,\ldots,z_m$ variables by values $b_1,\ldots,b_m$.

Note must however be taken that a path to b and path to b' do not interfere—because they have different "locations." Hence in particular, taking weighted sums of $p_C^L(a,B)$ is not the same as measuring outcomes in B. One needs to sum them for all $b \in B$. We will fix this issue by proving a parallel theorem for the acceptance probability. We state it here just for circuits of gates whose entries are multiples of i:

Theorem 2. *Let C be a circuit of m qubits and s balanced gates of minphase $\pi/2$ and maximum arity $r \le m$ qubits. Then we can efficiently build a Boolean formula ψ of size $O(m + s2^{2r})$ in variables $\vec{v},\vec{w},\vec{x},\vec{y},\vec{z}$ and find a constant R such that for all $a \in \{0,1\}^m$ and cylinders $B \subseteq \{0,1\}^m$:*

$$\sum_{b\in B} |\langle\, a \,|\, C \,|\, b \,\rangle|^2 = \frac{1}{R^2}(\#sat(\psi') - \#sat(\psi'')), \tag{1}$$

where ψ' and ψ'' are projections of ψ depending on B. Moreover, for every assignment (a,c) to $\vec{x},\vec{y}$ there is exactly one completion to an assignment that satisfies ψ' or ψ'' (never both) and it can be found in $O(s)$ time.

It is best to prove this after gaining a concrete understanding of the efficiency issues for the cases $K = 2, 4, 8$ and the motivation for sampling and uniform generation. The next section segregates the variables $\vec{y}$ of the general case into *Hadamard variables* $y_1,\ldots,y_h$ and other variables $v_1,\ldots,v_{s-h}$, whence "$\vec{v}$" in the above statement.

3 Binary Case

In this section, C is a circuit of Hadamard and Toffoli—and optionally NOT and CNOT—gates only. Then the nonzero entries are ± 1 ignoring factors of $\sqrt{2}$ and so the resulting values $e^{i\theta}$ are likewise ± 1. Paths giving $+1$ are *positive paths* and those giving -1 are *negative paths*. For any origin $a \in \{0,1\}^m$ and terminus $b \in \{0,1\}^m$ their numbers are denoted by $p_C^+(a,b)$ and $p_C^-(a,b)$, respectively. If $h \le s$ is the number of Hadamard gates then the amplitude of obtaining b as output by $C(a)$ is given by

$$\langle\, a \,|\, C \,|\, b \,\rangle = \frac{p_C^+(a,b) - p_C^-(a,b)}{2^{h/2}}.$$

This motivates the specific form of our first "concrete" main theorem in the binary case. For a cylinder B defined by $I = \{i_1,\ldots,i_r\}$ and c, let $\phi[\vec{z} \in B]$ stand for the substitution $z_{i_1} = c_1,\ldots,z_{i_r} = c_r$.

Theorem 3. *Given any m-qubit circuit C of h Hadamard gates and $s-h$ Toffoli and CNOT gates, input $a \in \{0,1\}^m$, and cylinder $B \subseteq \{0,1\}^m$, we can construct a Boolean formula ϕ_C of size $O(s+m)$ in conjunctive normal form with variables $y_1, \ldots, y_h, v_1, \ldots, v_{s-h}, w_, \ldots, w_h$ together with $x_1, \ldots, x_m$ and $z_1, \ldots, z_m$ such that*

$$\begin{aligned} p_C^+(a,B) &= \#sat(\phi[\vec{x} = a, \vec{z} \in B, w_h = 0]) \\ p_C^-(a,B) &= \#sat(\phi[\vec{x} = a, \vec{z} \in B, w_h = 1]). \end{aligned}$$

Moreover, no two satisfying assignments agree on $y_1, \ldots, y_h$.

Proof. We start with variables x_i, letting "u_i" initially refer to x_i on each line, and start with the equation $w_0 = 0$, i.e., $\bar{w}_0$. We let ℓ run from 1 to h this time, not 1 to s.

1. For each Hadamard gate on line i, increment ℓ, allocate fresh variables w_ℓ and y_ℓ, and conjoin the equation
$$(w_\ell = w_{\ell-1} \oplus (u_i \wedge y_\ell)).$$
To set up the next stage we note that "u_i" now refers to y_ℓ.

2. For each Toffoli gate with sources i, j and target k, increment o, allocate a fresh variable v_o and conjoin the equation
$$(v_o = u_k \oplus (u_i \wedge u_j)).$$
Now "u_k" refers to v_o.

3. For a CNOT gate with source i and target k, we conjoin $(v_o = u_k \oplus u_i)$ instead. Note this is the same as fixing $u_j = 1$ in the Toffoli case.

After placing the last gate, we conjoin the output-equating clauses $(u_i = z_i)$ for each qubit line i. Note again that in the case $h = 0$ we have $w_h = w_0$, and so $\phi_C[w_h = 1]$ is unsatisfiable—in keeping with there being no negative paths. The rest of the correctness logic follows as in the proof of Theorem 1 (or see the specialized proof in the Appendix).

A special beauty of this construction is that the differences among the three types of quantum gate are expressed by the fourth variable in the new equation being fresh, used, or fixed. The difference between $w_{\ell-1}$ and u_k also matters, of course, but both are existing variables.

To finish the proof of Theorem 3, we can use either of the following conversions to CNF. We can convert to 4CNF without introducing any more variables by applying to each equation the conversion

$$\begin{aligned} (q = p \oplus (u \wedge y)) &\equiv (\bar{u} \to p = q) \wedge (\bar{y} \to p = q) \wedge ((u \wedge y) \to p \neq q) \\ &\equiv (u \vee p \vee \bar{q}) \wedge (u \vee \bar{p} \vee q) \wedge (y \vee p \vee \bar{q}) \wedge (y \vee \bar{p} \vee q) \\ &\quad \wedge (\bar{u} \vee \bar{y} \vee p \vee q) \wedge (\bar{u} \vee \bar{y} \vee \bar{p} \vee \bar{q}). \end{aligned}$$

To obtain 3CNF we need to introduce a new variable t and equation $t = u \wedge y$. Doing so does not increase the number of satisfying assignments. The clauses thus obtained are:

$$\begin{aligned} (q = p \oplus t) \wedge (t = u \wedge y) &\equiv (\bar{t} \to p = q) \wedge (t \to p \neq q) \wedge (\bar{u} \to \bar{t}) \wedge (\bar{y} \to \bar{t} \wedge ((u \wedge y) \to t) \\ &\equiv (t \vee p \vee \bar{q}) \wedge (t \vee \bar{p} \vee q) \wedge (\bar{t} \vee p \vee q) \wedge (\bar{t} \vee \bar{p} \vee \bar{q}) \\ &\quad \wedge (u \vee \bar{t}) \wedge (y \vee \bar{t}) \wedge (\bar{u} \vee \bar{y} \vee t). \end{aligned}$$

The end equations $u_i = z_i$ become clause pairs $(u_i \vee \bar{z}_i) \wedge (\bar{u}_i \vee z_i)$. Overall, if C has m qubits, h Hadamard gates, and $s-h$ Toffoli plus CNOT gates, then the 4CNF formula has:

- h Hadamard variables $y_1, \ldots, y_h$;
- $h+1$ indicator variables $w_0, \ldots, w_h$;
- $s-h$ line variables $v_1, \ldots, v_{s-h}$;
- m input variables $x_1, \ldots, x_m$—which, however, are substituted for when presenting any input $a \in \{0,1\}^m$; and
- up to m output variables $z_1, \ldots, z_m$ per the discussion of cylinders above.

These variables form $6h + 6t = 6s$ 3-clauses and 4-clauses, plus the 1-clause $\bar{w}_0$ plus up to $2m$ output clauses, making $O(s+m)$ clauses in all. The 3CNF formula adds s more variables and has $7s$ clauses besides $(\bar{w}_0)$ and the output ones. As observed above, the formulas meet the requirements of Theorem 3 for any cylinder B. □

The one-qubit *Pauli gates* are defined as follows—note that X is the NOT gate and we have multiplied $\mathsf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ by i:

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathsf{iY} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

- To emulate X on line i, pro-forma we introduce a new variable v_o. The phase cannot change, so we have no new w_ℓ equation, but we need to rule out the two 0s in the matrix. Thus we add clauses $\neg(\bar{u}_i \wedge \bar{v}_o)$ and $\neg(u_i \wedge v_o)$. They amount to the single equation clause $(v_o = \bar{u}_i)$.

  As a shortcut we may just use $\bar{u}_i$ in place of u_i for the next gate involving line i (or the final output equation). In all of the above constructions and reasoning it is fine for "u_i" to be a negated variable. Thus the number of equation clauses may be less than the number of gates.

- To emulate Z on line i we can leave u_i alone since it is diagonal, but we need to flip the current indicator variable w_ℓ if $u_i = 1$. Thus we add the new variable w_ℓ and equation $w_\ell = u_i \oplus w_{\ell-1}$. Reflecting that the Y gate is deterministic, there is no change to the number of satisfying assignments—but now the number of phase-indicator variables will be greater than $h+1$.

- To emulate iY, we allocate v_o. We have to be mindful that u_i indexes the columns, so the -1 entry corresponds to $(\bar{u}_i \wedge v_o)$, To eliminate the 0s we do as above for X. Thus we get two new clauses:

  $$(w_\ell = w_{\ell-1} \oplus (\bar{u}_i \wedge v_o)) \wedge (v_o = \bar{u}_i).$$

  Thus the number of equations can be more than the number of gates. We can, however, employ the shortcut of making $\bar{u}_i$ become the location on line i and avoid introducing v_o, simplifying the phase equation to $(w_\ell = w_{\ell-1} \oplus \bar{u}_i)$. In either event, the normalizing constant R does not change,

As a check, note $\mathsf{iY} = \mathsf{ZX}$. In the order of presenting gates left-to-right in the circuit, the X comes first. Suppose we take the shortcut option. Then we next use $\bar{u}_i$ in place of "u_i" in the

description for Z. We thus get the new phase equation $w_\ell = \bar{u}_i \oplus w_{\ell-1}$, which agrees with the shortcut option for iY.

To emulate the phase gate $\mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$, however, we need to augment the series of indicator variables to handle non-integral multiples of π.

4 Gates With Other Phases

Most of the other common quantum gates conform to the original meaning of *bit* as an eighth of the Spanish dollar. So we take $k = 3$ and $K = 8$ to encode gates with entries that are powers of $\omega = \sqrt{i}$. The phases of paths can be identified with the integers modulo 8. We rename $w_{j,\ell}$ to the following three series of variables indexed only by ℓ:

- p_ℓ tracks whether the phase belongs to $\{0, 1, 2, 3\}$ or to $\{4, 5, 6, 7\}$;
- q_ℓ tracks whether it belongs to $\{0, 1, 4, 5\}$ or to $\{2, 3, 6, 7\}$;
- r_ℓ tracks whether the phase is even or odd.

The initial equations are $(p_0 = 0)$, $(q_0 = 0)$, and $(r_0 = 0)$, that is, singleton clauses $\bar{p}_0 \wedge \bar{q}_0 \wedge \bar{r}_0$. The formulas $\phi_j = \phi_C[W_s = j]$ are now obtained by substituting the bits of j for the respective final variables p_s, q_s, r_s. Again with $s = 0$ this will make $\phi_0 = \top$ when $a = b$ and $\phi_j = \bot$ for $j \neq 0$. When $s \geq 1$ it is OK to substitute 0 for p_0, q_0, r_0 to simplify the formula. Here are the rules for certain gates:

- $\mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$: No new variable v_o and no change to u_i since this is diagonal, but the phase increments by 2 modulo 8 when $u_i = 1$. So we add the new phase variables p_ℓ, q_ℓ with equations

$$\begin{aligned} q_\ell &= q_{\ell-1} \oplus u_i \\ p_\ell &= p_{\ell-1} \oplus (u_i \wedge q_{\ell-1}). \end{aligned}$$

- $\mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & \omega \end{bmatrix}$: This says to increment the phase modulo 8 if $u_i = 1$ else leave it unchanged. The equations are:

$$\begin{aligned} r_\ell &= r_{\ell-1} \oplus u_i \\ q_\ell &= q_{\ell-1} \oplus (r_{\ell-1} \wedge u_i \\ p_\ell &= p_{\ell-1} \oplus (q_{\ell-1} \wedge r_{\ell-1} \wedge u_i), \end{aligned}$$

The last equation expresses that there is a carry from the 1s place all the way to the 4s place (modulo 8). It goes outside the XOR-of-AND form of previous equations, but could be broken down with an extra variable v as $(p_\ell = p_{\ell-1} \oplus (u_i \wedge v)) \wedge (v = (q_{\ell-1} \wedge r_{\ell-1}))$. For S the parity does not change so there is no need to introduce r_ℓ, though no harm either in adding the equation $r_\ell = r_{\ell-1}$. The same goes for the controlled-S gate.

- $\mathsf{CS} = \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{S} \end{bmatrix}$ with source i and target j: Since this is diagonal, no new line variables are needed and u_i and u_j remain the markers on their lines. The new phase variables and clauses, again with no change to $r_{\ell-1}$, are:

$$\begin{aligned} q_\ell &= q_{\ell-1} \oplus (u_i \wedge u_j) \\ p_\ell &= p_{\ell-1} \oplus (u_i \wedge u_j \wedge q_{\ell-1}). \end{aligned}$$

- $\mathsf{R}_{\mathsf{z}\theta} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$ with $\theta = \pi/2$, giving $\begin{bmatrix} \bar{\omega} & 0 \\ 0 & \omega \end{bmatrix}$: The direct approach gives

$$\begin{aligned} r_\ell &= \neg r_{\ell-1}, \\ q_\ell &= q_{\ell-1} \oplus (r_{\ell-1} \oplus u_i) \\ p_\ell &= p_{\ell-1} \oplus ((u_i \wedge q_{\ell-1} \wedge r_{\ell-1}) \vee (\bar{u}_i \wedge \bar{q}_{\ell-1} \wedge \bar{r}_{\ell-1})), \end{aligned}$$

  with no change to u_i on the line. The equation for p_ℓ is annoying to convert into CNF. There are two alternatives. First, $\mathsf{R}_{\mathsf{z}\pi/2}$ is just the scalar multiple of S by $\bar{\omega}$, so we could ignore the scalar and do as for S. We would lose the literal correspondence between the values of (p_ℓ, q_ℓ, r_ℓ) and the phase, but the acceptance amplitudes would not change. However, we can accommodate the scalar by wriing equations involving just those and $(p_{\ell-1}, q_{\ell-1}, r_{\ell-1})$:

$$\begin{aligned} r_\ell &= \neg r_{\ell-1}, \\ q_\ell &= q_{\ell-1} \oplus \bar{r}_{\ell-1}, \\ p_\ell &= p_{\ell-1} \oplus (\bar{q}_{\ell-1} \wedge \bar{r}_{\ell-1}). \end{aligned}$$

  These equations all stay within the conversion to CNF given above, but taking two steps introduces an extra q_ℓ and r_ℓ variable.

- $\mathsf{V} = \sqrt{\mathsf{X}} = \frac{1}{\sqrt{2}} \begin{bmatrix} \omega & \bar{\omega} \\ \bar{\omega} & \omega \end{bmatrix}$: As written this is a nondeterministic gate, so we allocate a new line variable y_o which is *free* and becomes the "new u_i" in subsequent stages. Coding it directly has a similar annoyance factor, but we can use $\mathsf{V} = \bar{\omega} \cdot \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} i & 1 \\ 1 & i \end{bmatrix}$. The parity r_ℓ flips if we bother with the $\bar{\omega}$ factor but otherwise does not change. The other equations are:

$$\begin{aligned} q_\ell &= q_{\ell-1} \oplus (y_o = u_i) \\ p_\ell &= p_{\ell-1} \oplus (q_{\ell-1} \wedge (y_o = u_i)). \end{aligned}$$

  The first equation is equivalent to $q_\ell \oplus q_{\ell-1} \oplus y_o \oplus u_i$; as an XOR expression it has a 4CNF of eight clauses. The seond is equivalent to

$$(q_{\ell-1} \vee p_\ell \vee \bar{p}_{\ell-1}) \wedge (q_{\ell-1} \vee \bar{p}_\ell \vee p_{\ell-1}) \wedge (\bar{q}_{\ell-1} \vee F),$$

  where $F = p_\ell \oplus p_{\ell-1} \oplus y_o \oplus u_i$ is again an XOR of four literals. Hence we get a 5CNF of 2+8 clauses unless we introduce extra variables. The constant R goes up by a factor of 2, not just $\sqrt{2}$.

- $\mathsf{CV} = \begin{bmatrix} \mathsf{I} & 0 \\ 0 & \mathsf{V} \end{bmatrix}$: This 2-qubit gate is famous for combining with CNOT to simulate the Toffoli gate [BBC+95]. We can emulate it by replacing "u_i" in the expressions for V by $(u_i \wedge u_j)$, making y_o the new location on the target line j and leaving u_i alone on i. Conceptually simpler is to use the equation

$$\mathsf{CV} = (\mathsf{I} \otimes \mathsf{H}) \cdot \mathsf{CS} \cdot (\mathsf{I} \otimes \mathsf{H}),$$

that is, sandwiching CS between two Hadamard gates on line j.

Noting also that $\mathsf{CV} \cdot \mathsf{CV} = \mathsf{CNOT}$, it follows that CS and H form a universal set. It is interesting to compare the formulas obtained via various equivalences of this kind, especially in regard to the numbers of free variables added. The way parity formulas crop up in these equations enhances our interest in the construction in the Appendix, in case $\#sat$ solvers tailored to XOR form arise. The phase variables in all cases retain the property that upon setting the *free* variables $y_1, \ldots, y_h$ for Hadamard gates and any y_o for other nondeterministic gates as above, the values of all $p_{|ell}, q_\ell, r_\ell$ are forced.

To emulate $\mathsf{R_{z\theta}}$ or $T_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$ for finer angles $\theta = \pi/2^{K-1}$, we would add more series of phase-tracking variables mod 16, mod 32, and so forth. The sizes of formulas remain polynomial in $k = \log_2(K)$, but the number of phase terms in the formula

$$\langle\, a \,|\, C \,|\, b \,\rangle = \frac{1}{R} \sum_{L=0}^{K-1} \#sat(\phi[\vec{w} = j, \vec{a} = a, \vec{b} = b])\omega^L = \frac{1}{R} \sum_z \sum_L \phi(L, a, b, z)\omega^L$$

becomes exponential in k. Here is where the ability to take *approximations* in samples over these sums while driving Shor's algorithm may come in. We bring these issues right to a head by discussing the quantum Fourier transform next.

5 Encoding the QFT and Tradeoffs

The *quantum Fourier transform* $\mathsf{QFT_n}$ on n qubits is represented (in the standard basis) by the ordinary $N \times N$ discrete Fourier matrix F_N where $N = 2^n$. It has entries

$$F_N[I, J] = \omega^{I \cdot J},$$

where $\omega = e^{-2\pi i/N}$. If the incoming phase is c then the new phase is $c' = c + I \cdot J$ modulo N. Hence the only equations we need to add are

$$W_\ell = W_{\ell-1} + I \cdot J \pmod{N}. \tag{2}$$

Here we have suites of n phase variables for $W_{\ell-1}$ and W_ℓ, n qubit line literals u_i coding J, and n fresh location variables $y_1, \ldots, y_n$ coding I. Since F_N is symmetric we could interchange I and J. Another thing to note is that the parity in multiples of $1/N$ changes if and only if the row and column index of F_N (numbered from zero) are both odd. If the qubit lines on which F_N is situated are labeled $u_1, \ldots, u_n$ in order, then the odd row indices are those with $u_n = 1$ and the odd column indices have $v_n = 1$ for the newly-introduced variable v_n replacing u_n on that line. It follows that the equation for the finest-phase variable is simply

$$w_{n,\ell} = w_{n,\ell-1} \oplus (u_n \wedge v_n).$$

The string relation (2) is known to have Boolean circuits of size $O(n \log^{O(1)}(n))$ via the Schönhage-Strassen integer multiplication algorithm [SS71] and its conversion to circuits. (Ironicaly, this uses F_{2n}.) We do not know how efficiently the formulas convert to CNF but with the introduction of extra variables it is reasonable to suppose we can obtain bounded CNF of roughly $O(n^2)$ size. Alternatively, one can use Karatsuba multiplication in time $O(n^{1+\epsilon})$ and expect to get bounded-CNF formulas of size not too much worse. Concrete quantum circuits have recently been presented by Markov and Saeedi [MS12].

In any event, the inclusion of $\mathsf{QFT_n}$ in a circuit still allows formulas ϕ_C of $n^{O(1)}$ size. Lurking, however, is the issue of having 2^n phases. We will combine two ideas to finesse this: approximating the output distribution of $\mathsf{QFT_n}$ and using a $\#sat$ oracle on subformulas to guide the sampling.

A second way to emulate $\mathsf{QFT_n}$ is to use quantum circuits of simpler gates. Among various recursions for $\mathsf{QFT_n}$ in terms of recursions and similar gates, the following may be best tailored to our scheme of emulation. The qubit n-cycle defined by $\mathsf{K_n}|\, a_1a_2a_3\cdots a_n\,\rangle = |\, a_2a_3\cdots a_na_1\,\rangle$ and its inverse can be effected by cycling the u_i labels, without introducing any variables or adding any clauses. For all N define D_N to be the diagonal matrix with entries

$$D_N[I,I] = e^{\pi iI/N},$$

noting the absence of the '2' before 'π' so that this runs through the first N powers of a primitive $2N$-th root of unity. Then we obtain the recursion

$$F_N = \frac{1}{\sqrt{2}} \begin{bmatrix} I & D_{N/2} \\ I & -D_{N/2} \end{bmatrix} \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} \mathsf{K}_n^{-1}.$$

The Boolean equations for $D_{N/2}$ are much easier to handle than those for F_N directly because they involve only adding I modulo N to the phase, not multiplying. The identity matrices I here are $N/2 \times N/2$. Unrolling the recursion gives n stages, each of $O(n)$ formula size. Conversion to bounded-arity CNF and the presence of 2^n phases remain issues.

6 Sampling and Uniform Generation

To discuss sampling, first we review the classical way to sample satisfying assignments of a Boolean formula $\phi(x_1, \ldots, x_n)$ uniformly at random. If we try to do this with a SAT decision solver, then we might try the following: Ask whether $\phi_0 = \phi[x_1 = 0]$ and $\phi_1 = \phi[x_1 = 1]$ are satisfiable. If both are, then flip a coin to give the value of x_1, else just one is satisfiable and we must set x_1 accordingly. The flaw is that this does not give uniform distribution over the set S of satisfying assignments. There may be just one member a with $x_1 = 0$ and many with $x_1 = 1$, yet this process will give a with probability 0.5. It is in fact strongly believed that uniform sampling from S cannot be done with a decision solver, because that would collapse the complexity class $\#\mathsf{P}$ and hence the polynomial hierarchy down to P^{NP}.

With a $\#sat$ solver, however, we can calculate not only $|S|$ but also its breakdown according to the trial value of x_1. We can thus carry out the classical reduction from counting to uniform generation:

- Using two calls to $\#sat$, compute $|S_0| = |\{\, x \in S : x_1 = 0 \,\}|$ and $|S_1| = |\{\, x \in S : x_1 = 1 \,\}|$.
- Set $x_1 = 0$ with probability $|S_0|/|S|$ and $x_1 = 1$ otherwise.

- Substitute the value of x_1 into ϕ and recurse on x_2 and so on.

In the quantum case we could do this on $\phi = \phi_C$ to sample uniformly from S, but as remarked above in the binary case, this alone gives little information insofar as assignments for positive and negative paths are nearly evenly distributed. The *quantum sampling* task is to generate outputs b—belonging to $\{0,1\}^m$ or some (cylindrical) subset B thereof—according to the probability distribution $|\langle a \,|\, C \,|\, b \rangle|^2$. A naive attempt to emulate the above process—in the binary case $K = 2$ with h Hadamard gates—could go like this:

1. After substituting $\vec{x} = a$ in ϕ_C, substitute only $b_1 = 0$, that is, $z_1 = 0$, to create ϕ^0. Leave z_i for $i \geq 2$ alone.
2. Use the $\#sat$ oracle to compute $s_0^+ = \#sat(\phi^0[w_h = 0])$ and $s_0^- = \#sat(\phi_0[w_h = 1])$.
3. Fix $b_1 = z_1 = 0$ and $\phi = \phi^0$ with probability $p_0 = \frac{1}{2^h}(s_0^+ - s_0^-)^2$, else fix $b_1 = z_1 = 1$ and $\phi = \phi^1$.
4. Repeat the above steps for z_2, then z_3, and so on to the end.

This fails, however, because s_0^+ is the number of all positive paths from a to elements of $B_0 = \{\, b \in \{0,1\}^m : b_1 = 0 \,\}$, likewise s_0^- the number of all such negative paths. The probability of the output landing in B—that is, of measuring 0 in the first qubit—is not $(s_0^+ - s_0^-)^2$ but rather the sum of the squared differences for each individual $b \in B$, divided by 2^h.

We need a theorem to maintain these squared differences that preserves the cylindrical structure. The proof is a natural outgrowth of the ideas of Theorem 3 and Section 4. We state and prove the binary case first.

Theorem 4. *Let C be a circuit of m qubits, h Hadamard gates, and $s - h$ Toffoli and CNOT gates. Then we can efficiently build a Boolean formula ψ_C of size $O(m+s)$ in variables $\vec{v}, \vec{v'}, \vec{w}, \vec{w'}, \vec{x}, \vec{y}, \vec{y'}, \vec{z}$ and find a constant R such that for all $a \in \{0,1\}^m$ and cylinders $B \subseteq \{0,1\}^m$:*

$$\begin{aligned} \Pr[C(a) \in B] &= \sum_{b \in B} |\langle a \,|\, C \,|\, b \rangle|^2 \\ &= \frac{1}{2^h}(\#sat(\psi_C[\vec{x} = a, \vec{z} = b] \wedge w_h = w'_h) - \#sat(\psi_C[\vec{x} = a, \vec{z} = b] \wedge w_h \neq w'_h). \end{aligned}$$

Moreover, for every assignment (a, c, c') to $\vec{x}, \vec{y}, \vec{y'}$ there is exactly one completion to an assignment that satisfies ψ_C and it can be found in $O(s)$ time.

Proof. Take ϕ_C from Theorem 3 and make ψ_C from two copies of ϕ_C. The second copy shares the input and output variables and shares the initialization $w_0 = 0$ but has fresh Hadamard variables $y'_1, \ldots, y'_h$, fresh phase variables $w'_1, \ldots, w'_h$, and fresh line variables $v'_1, \ldots, v'_{s-h}$ from the non-Hadamard gates. By the analysis used to prove Theorem 3, for any $a, b \in \{0,1\}^m$:

- The number of satisfying assignments with $\vec{x} = a, \vec{z} = b$ that set $w_h = 0$ and that set $w'_h = 0$ equals $p_C^+(a,b)^2$.
- The number of satisfying assignments with $\vec{x} = a, \vec{z} = b$ that set $w_h = 1$ and that set $w'_h = 1$ equals $p_C^-(a,b)^2$.

- The number of satisfying assignments with $\vec{x} = a, \vec{z} = b$ that set w_h and w'_h differently equals $2p^+_C(a,b) \cdot p^-_C(a,b)$.

Equation (3) follows because we are summing $(p^+_C(a,b) - p^-_C(a,b))^2$ over all $b \in B$ and the denominator 2^h is the same for all terms. The previous proof analysis also establishes that every assignment (a, c, c') forces values of the line variables and phase variables in each copy and that the evaluation remains in $O(s)$ time. □

Now we can make the uniform generation procedure work by using $\psi_0 = \psi_C[z_1 = 0]$ in place of ϕ_0. We need two calls to the $\#sat$ oracle to evaluate $\#sat(\psi_0 \wedge w_h = w'_h)$ and $\#sat(\psi_0 \wedge w_h \neq w'_h)$. Their difference over 2^h gives the correct probability because by summing over assignments to $z_2, \ldots, z_m$ we are summing (3) over $b \in B_0$. We do not need to make separate calls for the case $z_1 = 1$. Thus the efficiency in the number of oracle calls is the same as in the classical iterative reduction. The resulting binary string b is generated with the same probability distribution $\mathcal{D}_C$ as measuring all registers of $C(a)$ would give, and the time needed is $O(mhT_{2s+m})$ where T_n is the time for calls to the $\#sat$ oracle on an n-variable formula.

The main pain is that the doubling-up of the previous ϕ_C changes the number of variables after substitutions from $s+m-1$ to $2s+m$ (the extra "+1" comes from w_h being equated to w'_h rather than fixed). There is, however, a sense in which the doubling would wind up being paid anyway. Suppose C is a decision circuit for inputs of length n. Standard definitions of acceptance by quantum circuits tend to say either that qubit 1 is measured or qubit $n+1$ is measured, with the binary outcome $g(x) = 1$ for acceptance, 0 for rejection. Let's say it is qubit $n+1$. If we say nothing more and work with ϕ_C, then we are working with $B_1 = \{\, b \in \{\,0,1\,\}^{n+1} : b_{n+1} = 1 \,\}$ and have the above-described problem that $\#sat(\phi_C[z_{n+1} = 1, \ldots])$ does not give what's needed.

The standard answer is to apply the so-called "compute-uncompute trick" by designing the bulk of C to operate on qubits $1, \ldots, n$, then use a CNOT gate to place the answer on line $n+1$, and then put a reversed and conjugated copy of the gates of C on the first n lines. The resulting circuit $C' = C^* \circ \mathsf{CNOT} \circ C$ maps $(a, 0)$ for any input $a \in \{\,0,1\,\}^n$ to $(a, g(a))$. Thus we have $b = a1$ as the unique target for $\langle\, a \,|\, C' \,|\, b \,\rangle$. Going from C to C', however, requires the same increase in fresh Hadamard and line variables as going to ψ_C. Indeed, the beautiful point is that $\phi_{C'}$ and ψ_C are virtually identical.

For $K = 4$ we again get a difference of two calls to $\#sat$:

Proof of Theorem 2. Here by the constructions in Section 4 the formula ϕ_C has variables p_h, q_h denoting the final phase, with $p_h = 0$ for 1 and i versus $p_h = 1$ for -1 and $-i$, and $q_h = 0$ for $1, -1$ versus $q_h = 1$ for $i, -i$. Again we make a copy ϕ'_C with final phase variables $p'_h < q'_h$. For a final state $\alpha = a + bi - c - di$ we have

$$|\alpha|^2 = (a^2 + b^2 + c^2 + d^2) - (2ac + 2bd).$$

The positive term is expressed by conjoining $(p'_h = p_h) \wedge (q'_h = q_h)$. The negative term is expressed by the combinations $(p_h q_h, p'_h q'_h) = (00, 10)$, $(10, 00)$, $(01, 11)$, or $(11, 01)$. The conjunction allowing exactly these combinations is $(p'_h \neq p_h) \wedge (q_h = q'_h)$. □

The case $K = 8$, however, brings a difference from theory when concreteness is desired. In theory the acceptance probability remains the difference between two #P functions, hence two invocations of $\#sat$, for any K. Indeed with $k = \log(K)$ allowed to grow polynomially in n it remains within the general closure theorem for the class GapP of [FFK94, FFL96, FR99]. However, with $K = 8$ the exponential sums have terms that are integral and terms multiplied by $\sqrt{2}$.

7 Driving Shor's Algorithm

Shor's algorithm to factor a given n-bit integer M starts by choosing $Q = 2^\ell$ where $\ell = 2n + 1$, so that $M^2 < Q < 2M^2$. It first guesses uniformly at random a number a such that $1 < a < M$. Presuming $\gcd(a, M) = 1$ (else we've found a factor of M by luck), it first sets up a deterministic circuit C_0 (using classical reversible gates) that maps any binary-encoded number $x < Q$ to $f_a(x) = a^x \pmod M$. More precisely, C_0 maps $x \cdot 0^\ell$ to the concatenation $x \cdot y$ where $y = f_a(x)$ as an ℓ-bit number. The execution of $C_0(x)$ using iterated squaring is actually the most time-consuming step by quantum reckoning and creates substantial overhead for our emulator. Here is the quantum part of the algorithm:

1. Prepend Hadamard gates on the first ℓ of 2ℓ qubit lines.
2. Then stick C_0 on the 2ℓ lines. This creates the *functional superposition*

$$\Phi = \frac{1}{\sqrt{Q}} \sum_x |\, x f_a(x) \,\rangle.$$

3. Apply QFT_ℓ to the first ℓ lines—that is. multiply by $F_Q \otimes I^{\otimes \ell}$.
4. Measure the first ℓ registers to get an output $b < Q$.

The algorithm then goes into deterministic steps that attempt to use b to find a *period* r such that $f_a(x) = f_a(x + r)$ for all x. With high probability, a correct r will cause a factor of M to tumble out. If the steps involving b fail, we start over again with another a'. The analysis shows that with high probability we need only $O(\log n)$ restarts.

The last step giving b can be emulated by exactly the sampling process of the last section. We can stop when z_ℓ is set—the remaining values and their probabilities will not matter.

Acknowledgments

References

[Ask14] Jonatan Asketorp. Attacking rsa moduli with sat solvers, 2014. Degree project.

[BBC+95] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P.W. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review Letters*, 52(5):3457–3467, 1995.

[BIS+16] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. https://arxiv.org/pdf/1608.00263.pdf, 2016.

[DHH+04] C. Dawson, H. Haselgrove, A. Hines, D. Mortimer, M. Nielsen, and T. Osborne. Quantum computing and polynomial equations over the finite field Z_2. *Quantum Information and Computation*, 5:102–112, 2004.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *J. Comp. Sys. Sci.*, 48:116–148, 1994.

[FFKL03] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle-builder's toolkit. *Inform. and Comp.*, 182:95–136, 2003.

[FFL96] S. Fenner, L. Fortnow, and L. Li. Gap-definability as a closure property. *Inform. and Comp.*, 130:1–17, 1996.

[FR98] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. In *Proc. 13th Annual IEEE Conference on Computational Complexity*, pages 202–206, 1998.

[FR99] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *J. Comp. Sys. Sci.*, 59:140–152, 1999.

[GS06] V. Gerdt and V. Severyanov. A software package to construct polynomial sets over Z_2 for determining the output of quantum computations. *Nuclear Instruments and Methods in Physics Research A*, 59:260–264, 2006.

[HSST] T. Häner, D. Steiger, M. Smelyanskiy, and M. Troyer. High performance emulation of quantum circuits. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, Utah, Nov. 2016*. IEEE press.

[LR14] R. Lipton and K. Regan. *Quantum Algorithms via Linear Algebra*. MIT Press, 2014.

[MS12] I. Markov and M. Saeedi. Constant-optimized quantum circuits for modular multiplication and exponentiation. *Quantum Information and Computation*, 12:361–394, 2012.

[NC00] M.A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[RC12] K. Regan and A. Chakrabarti. Quantum circuits, polynomials, and entanglement measures, 2012. Draft, in revision.

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, pages 124–134, 1994.

[SS71] A. Schönhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing* Arch. Elektron. Rechnen, 7:281–292, 1971.

[YB17] Henry Yuen and Joseph Babel. Tough SAT Project. https://toughsat.appspot.com/, 2017.

8 Appendix 1: Self-Contained Proof of Theorem 3

The following lemma acts as a "bridge" that may have separate application to SAT-solving in XOR form rather than CNF.

Lemma 1. *We can convert a quantum circuit C into a formula ψ_C of the form $\mu \wedge \eta$, where*

$$\mu = (\bar{p}_0 \oplus (u_1 \wedge y_1) \oplus (u_2 \wedge y_2) \oplus \cdots \oplus (u_h \wedge y_h)),$$

such that after substituting any values for a and b, $p_C^+(a,b) = \#sat(\mu \wedge \eta)[p_0 = 0]$ and $p_C^-(a,b) = \#sat(\mu \wedge \eta)[p_0 = 1]$. Here some u_i variables may coincide with a y_j variable, but the $\{ u_i \}$ are distinct, as are the $\{ y_j \}$.

The intuition is that CNOT and Toffoli gates have no effect on the signs of paths. Only when both input qubit and output qubit to a Hadamard gate are $|\,1\,\rangle$ does the path's sign change. The binary basis values of these bits are represented by the variable pairs u_j, y_j. Thus the path is positive if and only if it corresponds to an assignment that satisfies an even number of the terms $(u_j \wedge y_j)$ (including zero). It hence satisfies μ with $p_0 = 0$. Whereas, the path is negative if and only if it corresponds to an assignment that satisfies an odd number of the terms $(u_j \wedge y_j)$, and so satisfies μ with $p_0 = 1$, which is the same as negating μ. The contributions from CNOT and Toffoli gates go only into η and are "forced" by assignments to variables in μ so as to preserve input/output consistency. They are part of η.

Proof. Label the inputs with variables $x_1, \cdots, x_n$. If there are ancilla qubits, it is fine to continue labeling them $x_{n+1}, \cdots, x_m$ and substitute them with 0 later. Similarly label the outputs with variables $z_1, \cdots, z_n$, again using more if there are more qubits.

The first invariant of the construction is that at any stage each qubit line i has a "current variable" which we refer to indirectly as u_i. Initially each u_i is x_i. Upon declaring the circuit finished, we conjoin to η the equation clauses $(z_i = u_i)$ for each i. Initially we have

$$\psi_0 = \mu_0 \wedge \eta_0 = \bar{p}_0 \wedge \bigwedge_{i=0}^{m} (x_i = z_i) = \bar{p}_0 \wedge \bigwedge_{i=0}^{m} (\bar{x}_i \vee z_i) \wedge (x_i \vee \bar{z}_i).$$

Given any $a, b \in \{ 0,1 \}^m$, we have $p_0^+(a,b) = 1$ if $a = b$ and 0 otherwise, and we always have $p_0^+(a,b) = 0$ since there are no negative paths. The formula $\psi_0[x = a, z = b]$ equals $\perp$ (false) unless $a = b$, in which case it simplifies to $\bar{p}_0$. This has one satisfying assignment when $p_0 = 0$ and none with $p_0 = 1$. So Lemma 1 holds in this initial case. The induction goes as follows:

1. To add a Hadamard gate $\mathbf{H}_r$ on qubit line i, allocate a new variable y_r and define $\mu' = \mu \oplus (u_i \wedge y_r)$. Then y_r becomes the new "u_i" on line i for the next stage.

2. To add a CNOT gate with source on line i and target on j, allocate a new variable v_j, leave μ alone, and conjoin to η the equation $(v_j = u_j \oplus u_i)$. Then v_j becomes the new "u_j" for the next stage.

3. To add a Toffoli gate with sources on lines i and j and target on k, allocate a new variable v_k, leave μ alone, and conjoin to η the equation $(v_k = u_k \oplus (u_i \wedge u_j))$.

To prove correctness first in the case of a Toffoli gate, let any a, b be given, and consider C and μ_C and η_C before the gate was added to form C'. By inductive hypothesis, $p^+_C(a,b)$ equals the number of satisfying assignments to $(\mu \wedge \eta)[x = a, z = b]$ with $p_0 = 0$ and $p^-_C(a,b)$ the number with $p_0 = 1$. Then $\eta'[x = a, z = b]$ replaces the equation $(b_k = u_k)$ in η by

$$(b_k = v_k) \wedge (v_k = u_k \oplus (u_i \wedge u_j)).$$

Both have the equations $(b_i = u_i)$ and $(b_j = u_j)$. If $(b_i \wedge b_j)$ is false, then a positive path goes from a to b in C if and only if its unique extension in C' goes also to b, and ditto for a negative path. Hence

$$\begin{aligned} p^+_{C'}(a,b) &= p^+_C(a,b) = \#sat((\mu \wedge \eta)[x = a, z = b])[p_0 = 0] \\ p^-_{C'}(a,b) &= p^-_C(a,b) = \#sat((\mu \wedge \eta)[x = a, z = b])[p_0 = 1]. \end{aligned}$$

Any satisfying assignment of $(\mu \wedge \eta')$ must make $u_i \wedge u_j$ false and hence make $v_k = u_k$ true, so it yields a unique satisfying assignment of $(\mu \wedge \eta)$. Conversely, every satisfying assignment of $(\mu \wedge \eta)$ induces one of $(\mu \wedge \eta')$ with $v_k = u_k$. Hence the right-hand sides remain true with η' in place of η.

If $b_i = b_j = 1$, then let b' equal b with the bit b_k flipped. For every path that goes from a to b in C there is a unique path of the same sign that goes from a to b' in C' and vice-versa. Thus we have

$$\begin{aligned} p^+_{C'}(a,b) &= p^+_C(a,b') = \#sat((\mu \wedge \eta)[x = a, z = b'])[p_0 = 0], \\ p^-_{C'}(a,b) &= p^-_C(a,b') = \#sat((\mu \wedge \eta)[x = a, z = b'])[p_0 = 1]. \end{aligned}$$

Now an assignment that satisfies $\mu \wedge \eta$ with $u_k = b'_k$ extends to one of $\mu \wedge \eta'$ uniquely with $v_k = \neg u_k$ since it makes $(u_i \wedge u_j)$ true. Thus it makes $v_k = b_k$ back again and so satisfies $(\mu \wedge \eta)[x = a, z = b]$. The correspondence is 1-to-1 for the cases $p_0 = 0$ and $p_0 = 1$ separately (simply because p_0 does not appear in the clauses that differ between η and η'), so we obtain

$$\begin{aligned} p^+_{C'}(a,b) &= p^+_C(a,b') = \#sat((\mu \wedge \eta')[x = a, z = b])[p_0 = 0], \\ p^-_{C'}(a,b) &= p^-_C(a,b') = \#sat((\mu \wedge \eta')[x = a, z = b])[p_0 = 1]. \end{aligned}$$

in this case too, as needed to be proved. The case of a CNOT gate is similar and simpler.

In the case of a Hadamard gate on line i, now define b' to be b with bit i flipped. Let ψ abbreviate $\mu \wedge \eta$ with the substitutions $x_j = a_j$ and $z_k = b_k$ for all $k \neq i$, and similarly ψ' in relation to $\mu' \wedge \eta'$. So for instance, $\#sat((\mu \wedge \eta)[x = a, z = b])[p_0 = 0]$ is abbreviated by $\#sat(\psi[z_i = b_i])[p_0 = 0]$.

Let us first suppose $b_i = 0$, so $b'_i = 1$. Then positive paths from a to b in C split into two positive paths going to b and to b' in C', likewise negative paths. Whereas, positive paths from a to b' in C split into a positive path going to b in C' and a *negative* path going to b' in C', while negative paths from a to b' in C split in C' into a negative path to b and a positive path to b'. Focusing on the destination b, it follows that

$$\begin{aligned} p^+_{C'}(a,b) &= p^+_C(a,b) + p^+_C(a,b') = \#sat(\psi[z_i = b_i])[p_0 = 0] + \#sat(\psi[z_i = b'_i])[p_0 = 0], \\ p^-_{C'}(a,b) &= p^-_C(a,b) + p^-_C(a,b') = \#sat(\psi[z_i = b_i])[p_0 = 1] + \#sat(\psi[z_i = b'_i])[p_0 = 1]. \end{aligned}$$

Now ψ' differs from ψ only in having $\mu' = \mu \oplus (u_i \wedge y_i)$ and η' replacing $u_i = b_i$ in η by $y_i = b_i$ instead. Since $b_i = 0$, any satisfying assignment to ψ' sets $y_i = 0$ and hence makes $(u_i \wedge y_i)$

false. It follows that satisfying assignments to ψ' are in 1-to-1 correspondence with satisfying assignments to ψ that make $u_i = 0$ plus those that make $u_i = 1$ (whatever variable is being referred to as u_i), which are the same as those that make $z_i = 0$ in ψ plus those that make $z_i = 1$. Because this applies separately for $p_0 = 0$ and for $p_0 = 1$, we've shown

$$p^+_{C'}(a,b) = = \#sat(\psi'[z_i = 0])[p_0 = 0] = \#sat((\mu' \wedge \eta')[x = a, z = b])[p_0 = 0], \quad (3)$$
$$p^-_{C'}(a,b) = \#sat(\psi'[z_i = 0])[p_0 = 1] = \#sat((\mu' \wedge \eta')[x = a, z = b])[p_0 = 1]. \quad (4)$$

In the case $b_i = 1$ we have $b'_i = 0$. The previous reasoning about splitting paths applies with the roles of b and b' switched, so we get:

$$p^+_{C'}(a,b) = p^-_C(a,b) + p^+_C(a,b') = \#sat(\psi[z_i = b_i])[p_0 = 1] + \#sat(\psi[z_i = b'_i])[p_0 = 0],$$
$$p^-_{C'}(a,b) = p^+_C(a,b) + p^-_C(a,b') = \#sat(\psi[z_i = b_i])[p_0 = 0] + \#sat(\psi[z_i = b'_i])[p_0 = 1].$$

In both cases this is because the paths going straight across from b at the end of C to b at the end of C' flip sign. Satisfying assignments to $\psi'[z_i = b_i]$ now make $y_i = b_i = 1$, so $(u_i \wedge y_i)$ simplifies to u_i. Conditioned on $u_i = 0$, they satisfy ψ' if and only if they satisfy ψ. Conditioned on $u_i = 1 = b_i$, they satisfy ψ' with $p_0 = 0$ if and only if they satisfy ψ with $p_0 = 1$. The correspondence goes the other way as well: assignments satisfying ψ with $p_0 = u_i$ extend to satisfying assignments of ψ' with $p_0 = 0$ and those with $p_0 \neq u_i$ extend with $p_0 = 1$. Hence

$$\#sat(\psi')[p_0 = 0] = \#sat(\psi[z_i = b_i])[p_0 = 1] + \#sat(\psi[z_i = b'_i])[p_0 = 0],$$
$$\#sat(\psi')[p_0 = 1] = \#sat(\psi[z_i = b_i])[p_0 = 0] + \#sat(\psi[z_i = b'_i])[p_0 = 1].$$

This finally implies (3) and (4) in this case too, as needed to be proved. □

To give a direct proof of Theorem 3, no further argument about paths in circuits is needed—the rest only manipulates the Boolean formulas into conjunctive normal form by introducing the phase variables.

The final construction invariant is to maintain a variable p_r reflecting the current truth value of $\mu = \bar{p}_0 \oplus (u_1 \wedge y_1) \oplus \cdots \oplus (u_r \wedge y_r)$, which we abstract to $\bar{p}_0 \oplus \mu_1 \oplus \cdots \oplus \mu_r$. Note, incidentally, that our indexing convention $\phi_0 = \phi[p_0 = 0]$ for the positive paths and $p_0 = 1$ for negative reflects $(-1)^0 = 1$ and $(-1)^1 = e^{\pi i} = -1$, while an empty expression $\mu_1 \oplus \cdots \oplus \mu_r$ with $r = 0$ has the value *false*. Hence we actually make $\bar{p}_r$ equal the truth value of μ at stage r.

Lemma 2. *Given any formula $\mu \wedge \eta$ with $\mu = \bar{p}_0 \oplus \mu_1 \oplus \cdots \oplus \mu_r$, if we allocate new variables $p_1, \ldots, p_r$ and define*

$$\mu' = (\bar{p}_0 \oplus \mu_1 = \bar{p}_1) \wedge (\bar{p}_1 \oplus \mu_2 = \bar{p}_2) \wedge \cdots \wedge (\bar{p}_{r-1} \oplus \mu_r = \bar{p}_r) \wedge \bar{p}_r,$$

then $\#sat(\mu' \wedge \eta)[p_0 = 0] = \#sat(\mu \wedge \eta)[p_0 = 0]$ and $\#sat(\mu' \wedge \eta)[p_0 = 1] = \#sat(\mu \wedge \eta)[p_0 = 1]$.

Proof of Lemma 2 and Theorem 3. First suppose an assignment has $p_0 = 0$. Then it satisfies μ if and only if it makes an even number of the μ_i true. Since we must have $p_r = 0$, μ' simplifies to

$$(1 \oplus \mu_1 = \bar{p}_1) \wedge (\bar{p}_1 \oplus \mu_2 = \bar{p}_2) \wedge \cdots \wedge (\bar{p}_{r-1} \oplus \mu_r = 1). \quad (5)$$

This is satisfiable if and only if an even number of the μ_i are satisfied, and the values of $p_1, \ldots, p_{r-1}$ are forced. Going the other way, an assignment that satisfies μ' with $p_0 = 0$ must satisfy (5) with an even number of μ_i made true, hence it satisfies μ with $p_0 = 0$. Note that this

folds in the above reasoning for the case $r = 0$. Since η involves none of the variables $p_1, \ldots, p_{r-1}$ (nor p_0 nor p_r, in fact), its presence does not affect the 1-to-1 correspondence.

If the assignment has $p_0 = 1$ then it satisfies μ if and only if it makes an odd number of the μ_i true. Then μ' simplifies to

$$(0 \oplus \mu_1 = \bar{p}_1) \wedge (\bar{p}_1 \oplus \mu_2 = \bar{p}_2) \wedge \cdots \wedge (\bar{p}_{r-1} \oplus \mu_r = 1), \tag{6}$$

which is likewise satisfiable—uniquely—if and only if an odd number of the μ_i are made true. Once again this respects the case $r = 0$ and the presence (or absence) of η. This also completes the proof of Theorem 3. □

The above argument goes through regardless of which signs we choose for $p_1, \ldots, p_r$, provided the sign of p_r is the same in the last equation and the last conjunct. We have chosen the signs all negative to facilitate one further observation: we can now flip them all to have positive sign. This changes the interpretation of the indices: now $\phi_0 \equiv \phi[p_0 = 0]$ corresponds to p_C^- rather than p_C^+. But since the final acceptance probability is a function of $(p^+ - p^-)^2$ even this difference does not matter to the simulation.