FURTHER EXTENSIONS OF CLIFFORD CIRCUITS AND THEIR CLASSICAL SIMULATION COMPLEXITIES

DAX ENSHAN KOH

Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

ABSTRACT. Extended Clifford circuits, such as those in [1], straddle the boundary between classical and quantum computational power. Whether such circuits are efficiently classically simulable seems to depend delicately on the ingredients of the circuits. While some combinations of ingredients lead to efficiently classically simulable circuits, other combinations that might just be slightly different, lead to circuits which are likely not. We extend the results of [1] by studying two further extensions of Clifford circuits. Firstly, we consider how the classical simulation complexity changes when we allow for more general measurements. Secondly, we investigate different notions of what it means to 'classically simulate' a quantum circuit. These further extensions give us 24 new combinations of ingredients compared to [1], and we give a complete classification of their classical simulation complexities. Our results provide more examples where seemingly modest changes to the ingredients of Clifford circuits lead to large changes in the classical simulation complexity under plausible complexity assumptions.

1. INTRODUCTION

Clifford circuits are an important class of circuits in quantum computation [2]. They have found numerous applications in quantum error correction [3], measurement-based quantum computation [4, 5] as well as quantum foundations [6, 7]. One of the central results about Clifford circuits is the Gottesman-Knill Theorem [2], which states that such circuits can be efficiently simulated on a classical computer, and hence do not provide a speedup over classical computation. But this is true only in a restricted setting – whether or not we can efficiently classically simulate such circuits depends delicately on the 'ingredients' of the circuit, for example, on the type of inputs we allow, whether or not intermediate measurements are adaptive, the number of output lines, and even on the precise notion of what it means to *efficiently simulate* a circuit. These cases were considered in [1], who showed that many of these 'extended' Clifford circuits are in fact not classically simulable under plausible complexity assumptions.

One of the main motivations for studying extended Clifford circuits is that they shed light on the relationship between quantum and classical computational power. Are quantum computers more powerful than their classical counterparts? If so, what is the precise boundary between their powers? One approach to answering this question is to consider restricted models of quantum computation and study their classical simulation complexities. For example, suppose that we start with a restricted model $\mathcal{A}$ that is efficiently classically simulable. If adding ingredients $\mathcal{P}$ to $\mathcal{A}$ creates a new class that is universal for quantum computation, then we could regard $\mathcal{P}$ as an essential 'resource' for quantum computational power. Extended Clifford circuits, as a restricted model of quantum computation, are especially well-suited for this approach as they straddle the boundary between classical and quantum computational power. One could give many examples where adding a seemingly modest ingredient to an extended Clifford circuit changes it from being efficiently classically simulable to one that is likely not.

Understanding how the classical simulation complexity of extended Clifford circuits changes when various new ingredients are added is a central goal of this paper. In [1], the authors tabulate the simulation complexities of Clifford circuits with 16 different combinations of ingredients. In particular, they consider the different combinations of ingredients that arise from 4 binary choices: computational basis inputs vs product state inputs, single-line outputs vs multiple-line outputs, nonadaptive measurements vs adaptive measurements, and weak vs strong simulation. In this paper, we extend their results in two ways. First, we study how the classical simulation complexity changes when we employ a weaker notion of simulation than strong simulation, which we

Date: December 11, 2015.

call $\mathsf{STR}(n)$ simulation (such a notion seems incomparable with weak simulation). Second, we study how the classical simulation complexity changes when we allow for general product measurements (called OUT(PROD)) instead of just the computational basis measurements (called OUT(BITS)) that were considered in [1]. With these additional ingredients, the number of different combinations of ingredients grows to 40. We summarize the classical simulation complexities of each of these cases in Table 1.

2. Preliminary definitions and notations

We review the definitions introduced in [1]: the standard Pauli matrices are denoted by I, X, Y, Z, and an n-qubit Pauli operator is defined to be any operator of the form $P = i^k P_1 \otimes \ldots \otimes P_n$, where $k = 0, 1, 2, 3$ and each P_i is a Pauli matrix. The set of n-qubit Pauli operators forms a group $\mathcal{P}_n$, called the Pauli group. The n-qubit Clifford group $\mathcal{C}_n$ is defined to be the normalizer of the Pauli group $\mathcal{P}_n$ in the n-qubit unitary group $\mathcal{U}_n$, i.e. $\mathcal{C}_n = \{U \in \mathcal{U}_n | U P_n U^\dagger = P_n\}$. Any element of the Clifford group is called a *Clifford operation.* Clifford operations have an alternative characterization [8]: an n-qubit operator C is a Clifford operation if and only if it can be written as a circuit consisting of $O(n^2)$ gates from the following list: the Hadamard gate $H = 1/\sqrt{2}(X + Z)$, the phase gate $S = \mathrm{diag}(1, i)$, and the CNOT gate $\mathrm{CX}_{ab} = |0\rangle\langle 0|_a \otimes I_b + |1\rangle\langle 1|_a \otimes X_b$. Following the terminology in [1], we call these gates the *basic Clifford gates.* A *unitary Clifford circuit* is one that comprises only the basic Clifford gates. A *Clifford circuit* is one that consists of not just the basic Clifford gates but also single-qubit intermediate measurement gates in the computational basis.

We consider *Clifford computational tasks* of the following form:

(1) Start with an n-qubit pure input state $|\psi_{\mathrm{in}}\rangle$.
(2) Apply to $|\psi_{\mathrm{in}}\rangle$ a Clifford circuit B, which may be expressed as:
$$\begin{aligned} B(x_1, \ldots, x_K) &= C_K(x_1, \ldots, x_K) M_{i_K(x_1, \ldots, x_{k-1})}(x_K) \ldots \\ &\qquad C_2(x_1, x_2) M_{i_2(x_1)}(x_2) C_1(x_1) M_{i_1}(x_1) C_0, \end{aligned} \tag{1}$$
where each $C_i(x_1, \ldots, x_i)$ is a Clifford unitary circuit and $M_i(x)$ indicates a measurement on qubit line i with measurement result x. In general, B is taken to be an adaptive circuit, i.e. the ith unitary Clifford circuit C_i depends on previous measurement results $x_1, \ldots, x_i$. Let N denote the total number of gates in B. Assume that there are no extraneous qubits, so that $n = O(N)$.
(3) Measure all n qubit lines using a projection-valued measure $\{|\beta_{y_1,\ldots y_n}\rangle\langle\beta_{y_1,\ldots y_n}|\}_{y_1,\ldots,y_n}$, with measurement outcome $y_1 y_2 \ldots y_n \in \{0, 1\}^n$.

In this work, we restrict our attention to product state inputs and product state measurements, i.e.

(1) Inputs are $|\psi_{\mathrm{in}}\rangle = |\alpha_1\rangle|\alpha_2\rangle \ldots |\alpha_n\rangle$, where each $|\alpha_i\rangle \in \mathbb{C}^2$.
(2) Measurements directions are $|\beta_{y_1,\ldots y_n}\rangle = |\beta_1^{y_1}\rangle|\beta_2^{y_2}\rangle \ldots |\beta_n^{y_n}\rangle$, where each $|\beta_i^{y_i}\rangle \in \mathbb{C}^2$.

Note that for each i, by completeness, $|\beta_i^0\rangle\langle\beta_i^0| + |\beta_i^1\rangle\langle\beta_i^1| = I$. Hence, we need to just specify $\{|\beta_i^0\rangle\langle\beta_i^0|\}_i$ in order to completely specify the product state measurement. A description of the Clifford computational task is thus given by the three-tuple
$$T = (|\alpha\rangle, B, |\beta\rangle), \tag{2}$$
where $|\alpha\rangle = |\alpha_1\rangle|\alpha_2\rangle \ldots |\alpha_n\rangle$ is the initial state, B is the description of the Clifford circuit, and $|\beta\rangle = |\beta_1^0\rangle|\beta_2^0\rangle \ldots |\beta_n^0\rangle$ are the measurement directions.

Now, each product state input can be seen as arising from applying a product unitary to the computational basis states, i.e. there exist single-qubit unitary operators $V_1, \ldots, V_n$ such that $V_1 \otimes \ldots \otimes V_n |0 \ldots 0\rangle = |\alpha\rangle$. Likewise, every product state measurement operator can be seen as arising from applying a unitary operator followed by measuring in the computational basis. More precisely, a measurement in the direction $|\beta_1^{y_1}\rangle|\beta_2^{y_2}\rangle \ldots |\beta_n^{y_n}\rangle$ is equivalent to the application of a unitary operator $U_1^\dagger \otimes \ldots \otimes U_n^\dagger$ followed by a measurement in the computational basis, where the y_ith (with zero indexing) column of U_i is given by $U_i|y_i\rangle = |\beta_i^{y_i}\rangle$.

Hence, the Clifford computational tasks we consider are of the structure shown in Fig. 1. They may alternatively be represented by the 3-tuple
$$T = (\{V_i\}_{i=1}^n, B, \{U_i\}_{i=1}^n). \tag{3}$$
We will use the above two descriptions in Eqs. (2) and (3) of Clifford tasks interchangeably, and even allow for mixed descriptions, for example, $T = (|\alpha\rangle, B, \{U_i\}_{i=1}^n)$.

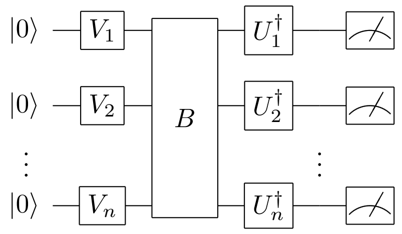


FIGURE 1. Circuit diagram for the Clifford computational tasks considered in this paper. The gates V_i and $U_i^\dagger$ are arbitrary single qubit unitaries, B is a Clifford circuit, the input state is the all-zero computational basis state, and the output measurement is performed in the computational basis.

We'll now write formal expressions for the probabilities of outcomes. For a computational task $T = (|\alpha\rangle, B, |\beta\rangle)$ and subset $I = \{i_1, \dots, i_s\} \subseteq [n]$, let $P_T^I(y_{i_1}, \dots, y_{i_n})$ be the marginal probability that the outputs $y_{i_1}, \dots, y_{i_s}$ are obtained for the lines $i_1, \dots, i_s$. Define $P_T(y_1, \dots, y_n) = P_T^{[n]}(y_1, \dots, y_n)$ to be the probability of the outcome $y_1 y_2 \cdots y_n$.

For the adaptive circuit described by Eq. (1), if the intermediate measurement results are $x_1 \dots x_K$, then the density operator of the final state is given by $B(x_1, \dots, x_K)[\rho_\alpha]$, where $\rho_\alpha = |\alpha\rangle\langle\alpha|$. We use the notation $C[\rho]$ to denote the state that results when we apply U to the density matrix ρ, i.e. $C[\rho] = C\rho C^\dagger$. The probability that the result $x_1 \dots x_K$ occurs is given by

$$p(x_1, \dots, x_K) = p(x_K|x_1, \dots, x_{K-1})p(x_{K-1}|x_1, \dots, x_{K-2}) \dots p(x_2|x_1)p(x_1),$$

where

$$p(x_j|x_1, \dots, x_{j-1}) = \operatorname{tr}\{|x_j\rangle\langle x_j|_{i_j} C_{j-1}(x_1, \dots, x_{j-1}) M_{i_{j-1}(x_1,\dots,x_{j-2})}(x_{j-1}) \cdots \quad (4)$$

$$\times C_1(x_1) M_{i_1}(x_1) C_0[\rho_\alpha]\}. \quad (5)$$

The final output state is then given by

$$B[\rho_\alpha] = \sum_{x_1 \dots x_K} p(x_1, \dots, x_K) B(x_1, \dots, x_K)[\rho_\alpha].$$

Hence, the outcome probabilities are given by

$$p_T(y_1, \dots, y_n) = \langle \beta_{y_1,\dots y_n} | B[\rho_\alpha] | \beta_{y_1,\dots y_n} \rangle,$$

and the marginal probabilities are given by

$$p_T^I(y_{i_1}, \dots, y_{i_n}) = \sum_{y_{k_1} \cdots y_{k_{n-s}}} p_T(y_1, \dots, y_n), \quad (6)$$

where $\{k_1, \dots, k_{n-s}\} = [n] - I$.

We consider the following 3 binary choices of ingredients:

(1) Inputs: IN(BITS) vs IN(PROD)
(2) Intermediate measurements: NONADAPT vs ADAPT
(3) Outputs: OUT(BITS) vs OUT(PROD)

The first two cases have been considered in [1]: IN(BITS) and IN(PROD) refer to having computational basis inputs and product state inputs respectively, while NONADAPT and ADAPT refer to nonadaptive and adaptive measurements respectively. Note that in [1], all the output measurements are performed in the computational basis (call this case OUT(BITS)). A natural extension to [1] would thus be to consider more general measurements. For the sake of symmetry with the inputs, we introduce the new ingredient OUT(PROD), which refers to product output measurements, i.e. when the U_i's in Eq. (3) are unrestricted. Note that we allow product state measurements only at the output – intermediate measurements are always single-qubit measurements in the computational basis.

These 3 binary choices lead to $2^3 = 8$ different subsets of Clifford computational tasks. Let $\nu \in \{$(IN(BITS), NONADAPT, OUT(BITS)), (IN(BITS), NONADAPT, OUT(PROD)), $\dots\}$ be one of these 8 subsets. We shall denote the subset of Clifford computational tasks corresponding to ν by $\mathcal{C}_\nu$. Note that unlike [1], we do not

$$\begin{tikzcd}
 & \mathsf{STR} \arrow[dl] \arrow[d] \arrow[dr] & \\
\begin{array}{c}\mathsf{WEAK} = \\ \mathsf{WEAK}(n)\end{array} \arrow[dr] & \mathsf{STR}(1) \arrow[d] & \mathsf{STR}(n) \arrow[dl] \\
 & \mathsf{WEAK}(1) &
\end{tikzcd}$$

Figure 2. Relationships between different notions of classical simulation of Clifford computational tasks.

include OUT(1) and OUT(MANY) as ingredients in our circuit. Instead, as stated above, we assume without loss of generality that all n qubit lines are measured. This is justified by the principle of implicit measurement, which states that any unterminated quantum wires at the end of the circuit can be assumed to be measured [8]. The number of output lines we simulate will be specified by the notion of simulation instead. We discuss these notions in the next section.

3. Notions of classical simulation

In [1], Jozsa and Van den Nest consider two notions of classical simulation, namely weak (WEAK) and strong (STR) simulation. A weak simulation involves providing a sample of the output distribution, while a strong simulation involves a calculation of the probability values of not just the joint distribution but also of all the marginal distributions. An immediate question that then arises is the following: what if we are required to calculate just the probability values for the joint distribution, but not of any of the marginals? Let's call such a simulation $\mathsf{STR}(n)$. How does $\mathsf{STR}(n)$ compare with STR and WEAK? How would the classical simulation complexity of Clifford circuits with various ingredients change if we used $\mathsf{STR}(n)$ as our notion of simulation instead?

To formalize these notions, we make the following definitions. Let $f(n)$ be either the constant function $f(n) = 1$ or the linear function $f(n) = n$ (in this paper, we restrict our attention to these cases, though one might certainly consider other functions f, like $f(n) = \log(n)$).

Definition 1. ($\mathsf{STR}(f(n))$) A $\mathsf{STR}(f(n))$ simulation of a subset of Clifford computational tasks $\mathcal{C}_\nu$ is a deterministic classical algorithm that on input $\langle T, I, y\rangle$, where $T \in \mathcal{C}_\nu$ is a task on n qubits, $I = \{i_1, \ldots, i_{f(n)}\} \subseteq [n]$ and $y_I = \{y_{i_1}, \ldots, y_{i_{f(n)}}\}$, outputs $p_T^I(y_{i_1}, \ldots, y_{i_{f(n)}})$.

Definition 2. ($\mathsf{WEAK}(f(n))$) A $\mathsf{WEAK}(f(n))$ simulation of a subset of Clifford computational tasks $\mathcal{C}_\nu$ is a randomized classical algorithm that on input $\langle T, I\rangle$, where $T \in \mathcal{C}_\nu$ is a task on n qubits and $I = \{i_1, \ldots, i_{f(n)}\} \subseteq [n]$, outputs $y_{i_1}, \ldots, y_{i_{f(n)}}$, with probability $p_T^I(y_{i_1}, \ldots, y_{i_{f(n)}})$.

STR and WEAK simulation are defined in exactly the same way, except that we place no restrictions on the size of the subset of output lines $|I|$ we simulate.

Let $\mathcal{S} \in \{\mathsf{STR}(n), \mathsf{STR}(1), \mathsf{STR}, \mathsf{WEAK}(n), \mathsf{WEAK}(1), \mathsf{WEAK}\}$ be one of these 6 notions of simulation. We define an $\mathcal{S}$-simulation of a subset of Clifford computational tasks $\mathcal{C}_\nu$ to be *efficient* if the simulation runs in $\mathsf{poly}(N)$-time, where N is the number of gates in the $\mathcal{C}_\nu$-circuit. Let $\mathsf{P}\mathcal{S}$ be the set of all $\mathcal{C}_\nu$ that have an efficient $\mathcal{S}$-simulation.

An immediate observation is that $\mathsf{PWEAK} = \mathsf{PWEAK}(n)$. The backward inclusion holds by definition, and the forward inclusion holds because we could sample from any subset I by just ignoring the qubit lines that are not in I. By definition, we also get the following inclusions: $\mathsf{PSTR} \subseteq \mathsf{PSTR}(1)$, $\mathsf{PSTR} \subseteq \mathsf{PSTR}(n)$ and $\mathsf{PWEAK} \subseteq \mathsf{PWEAK}(1)$. But how does weak simulation compare with strong simulation? From Proposition 1 of [9], it follows that $\mathsf{PSTR} \subseteq \mathsf{PWEAK}$, $\mathsf{PSTR}(1) \subseteq \mathsf{PWEAK}(1)$ and $\mathsf{PSTR}(n) \subseteq \mathsf{PWEAK}(1)$. It turns out that these are the only inclusions we know, unless plausible complexity assumptions like $\mathsf{P} \neq \mathsf{P}^{\#\mathsf{P}}$ turn out to be

			Weak		Strong		
			WEAK(1)	WEAK(n)	STR(1)	STR(n)	STR
OUT (BITS)	NON-ADAPT	IN (BITS)	P (i)	P (ii)	P (iii)	P (iv)	$\boxed{\mathsf{P}}$ (JV4)
		IN (PROD)	P (v)	$\boxed{\mathsf{PH}}$ (JV7)	P (JV1)	$\boxed{\#\mathsf{P}}$ (Thm 1)	#P (JV6)
	ADAPT	IN (BITS)	P (vi)	$\boxed{\mathsf{P}}$ (JV5)	$\boxed{\#\mathsf{P}}$ (JV2)	$\boxed{\#\mathsf{P}}$ (Thm 2)	#P (vii)
		IN (PROD)	$\boxed{\mathsf{QC}}$ (JV3)	QC (viii)	#P (ix)	#P (x)	#P (xi)
OUT (PROD)	NON-ADAPT	IN (BITS)	P (xii)	$\boxed{\mathsf{PH}}$ (Thm 3)	P (xiii)	$\boxed{\#\mathsf{P}}$ (Thm 4)	#P (xiv)
		IN (PROD)	P (xv)	PH (xvi)	$\boxed{\mathsf{P}}$ (Thm 5)	#P (xvii)	#P (xviii)
	ADAPT	IN (BITS)	$\boxed{\mathsf{P}}$ (Thm 6)	PH (xix)	#P (xx)	#P (xxi)	#P (xxii)
		IN (PROD)	QC (xxiii)	QC (xxiv)	#P (xxv)	#P (xxvi)	#P (xxvii)

TABLE 1. Classification of the classical simulation complexities of classes of Clifford circuits with different ingredients. P stands for efficiently classically simulable. #P stands for #P-hard. QC stands for QC-hard and PH stands for "if efficiently classically simulable, then the polynomial hierarchy collapses". The proofs of JV 1–7 are given in [1]. Theorems 1–6 concern cases not found in [1] and are our main results. (i)–(xxvii) are results that follow immediately from these theorems by using the rules in Appendix A. The 11 cases with boxed symbols are the core theorems, from which all other cases can be deduced using rules which we describe in Appendix A. These include all the main theorems JV–7 and Theorems 1–6, except JV1 and JV6, which turn out to be special cases of Theorem 5 and Theorem 1 respectively, using the rules in Appendix A.

false. One might wonder why, for example, we did not include the inclusion $\mathsf{PSTR}(n) \subseteq \mathsf{PSTR}(1)$ above. Well, we don't know if it's true! Computing such a marginal distribution directly from the joint distribution would involve summing an exponential number of terms, and unless $\mathsf{P} \neq \mathsf{P}^{\#\mathsf{P}}$, there seems to be no way around that. We summarize the relationships between the different notions in Figure 2.

4. Results and discussion

In Section 2, we introduced 3 binary choices of ingredients. In Section 3, we described 5 different notions of classical simulation. This gives a total of $2^3 \times 5 = 40$ different cases, whose classical simulation complexity we classify in Table 1. The entries of the table should be understood as follows: for a subset of computational tasks $\mathcal{C}_\nu$, and a notion of simulation $\mathcal{S}$,

- P (classically efficiently simulable) means that $\mathcal{C}_\nu \in \mathsf{P}\mathcal{S}$.
- #P (which stands for #P-hard) means that an efficient S-simulation of $\mathcal{C}_\nu$ would give rise to an efficient algorithm for the #P-complete problems.
- QC (which stands for quantum-computing universal) means that $\mathcal{C}_\nu$ is universal for quantum computation.
- PH means that an efficient S-simulation of $\mathcal{C}_\nu$ would imply a collapse of the polynomial hierarchy. (see [1] for more details)

Our main results are Theorems 1–6, whose proofs we present in Appendices B–G. Using the rules in Appendix A, these theorems, together with the results[1] JV 1–7 from [1], give a complete classification of the classical simulation complexities of all the 40 cases.

A few remarks are in order. First, we note that the entries in the last two columns of Table 1 are identical. This suggests that even though the notions STR(n) and STR(1) seem to be incomparable, for Clifford computational

[1]JV = Jozsa and Van den Nest [1]

tasks, $\mathsf{STR}(n)$-simulation appears to be harder to perform than $\mathsf{STR}(1)$-simulation. We note that Theorem 1, which generalizes (JV6), implies that being able to compute only the joint probabilities already suffices in enabling us to solve #P-hard problems; we do not require the full power of strong simulation.

Second, we note the symmetry between the inputs states and output measurements: for example, the 2nd and the 5th rows are identical, i.e. the simulation complexity remains the same whether product unitaries are applied at the beginning or at the end of the circuit. In (JV7), for example, the key to collapsing the polynomial hierarchy was that the magic state $|\pi/4\rangle = 1/\sqrt{2}(|0\rangle + e^{i\pi/4}|1\rangle)$ together with postselection can simulate the $T = \text{diag}(1, e^{i\pi/4})$ gate. In the proof of Theorem 3, however, we do not have magic state inputs at our disposal. Yet, it turns out that the T gate can also be simulated by what might be termed "magic measurements with postselection".

Third, Theorem 5 is a generalization of JV1. In fact, a stronger result can be similarly shown to be true: for any constant b, $\mathsf{STR}(b)$-simulation of OUT(PROD), NONADAPT, IN(PROD) can be performed efficiently. In [10], Aaronson and Gottesman present algorithms for simulating two separate cases: non-stabilizer initial states, and non-stabilizer gates. A consequence of their results is that it is efficient to simulate (in the $\mathsf{STR}(b)$-sense) nonadaptive tasks with either of the following ingredients: 1. product state inputs with computational basis measurements (which is the content of JV1). 2. computational basis inputs with product state measurements (which is the content of case xiii) – since this is equivalent to applying b single-qubit gates just before a computational basis measurement. Theorem 5 is slightly more general than either of these cases. Essentially, it combines product state inputs with product measurements and shows that the new task is still in $\mathsf{PSTR}(b)$.

5. Concluding remarks

We have shown how the classical simulation complexity of extended Clifford circuits changes when the ingredients in the circuit are varied. The cases which are efficiently classically simulable can be considered variants or extensions of the Gottesman-Knill Theorem. Various other ingredients besides those considered here have been studied in the literature, for example, mixed input states [10], states (as well as transformations and measurements) with positive Wigner representations [11, 12], and even non-commutative extensions like XS-stabilizer states [13]. The Gottesman-Knill Theorem has also been extended to continuous-variable quantum systems [14] as well as to normalizer circuits [15]. However, most of these extensions have been considered separately. It will be fruitful to study how the different combinations of these other ingredients affect the simulation complexity of the computational task.

Our discussion of extended Clifford circuits has been restricted to involve only exact simulation; it will be interesting to see how the classical simulation complexities of the various subsets $\mathcal{C}_\nu$ would change if we allowed for some error in the simulation. For example, we showed that an efficient exact $\mathsf{WEAK}(n)$ simulation of several combinations of ingredients would lead to a collapse of the polynomial hierarchy. Would the hierarchy still collapse if we allowed for error? If so, for what kinds of error would the result hold? These questions have been asked of other restricted models of quantum computation, like IQP circuits [16] and boson-sampling [17]. For the boson-sampling model, Aaronson and Arkhipov [17] show that even if the classical simulation were approximate or noisy, the polynomial hierarchy would still collapse, but only if certain unproven conjectures, like the *Permanent-of-Gaussians Conjecture*, and the *Permanent Anti-Concentration Conjecture* are true. It will be interesting to consider different notions of approximation or noise for Clifford computational tasks and ask what assumptions we will need in order to prove a polynomial-hierarchy collapse, or other hardness results.

Acknowledgements

I thank Anand Natarajan and Adam Bouland for helpful discussions on some of the proofs of the results in Table 1, and Scott Aaronson for teaching the wonderful class 6.S899 on Physics and Computation.

References

[1] R. Jozsa and M. Van den Nest, "Classical simulation complexity of extended clifford circuits," *Quantum Info. Comput.*, vol. 14, pp. 633–648, 2014.

[2] D. Gottesman, "The Heisenberg representation of quantum computers," *Talk at International Conference on Group Theoretic Methods in Physics*, arXiv: quant-ph/9807006 1998.

[3] D. Gottesman, *Stabilizer Codes and Quantum Error Correction.* PhD thesis, California Institute of Technology, 1997.

[4] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states," *Phys. Rev. A*, vol. 68, p. 022312, Aug 2003.

[5] R. Raussendorf and H. J. Briegel, "A one-way quantum computer," *Phys. Rev. Lett.*, vol. 86, pp. 5188–5191, May 2001.

[6] M. Pusey, "Stabilizer notation for Spekkens' toy theory," *Found Phys*, vol. 42, no. 688-708, 2012.

[7] R. W. Spekkens, "Evidence for the epistemic view of quantum states: A toy theory," *Phys. Rev. A*, vol. 75, p. 032110, Mar 2007.

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

[9] B. Terhal and D. DiVincenzo, "Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games," *Quant. Inf. Comp.*, vol. 4, no. 2, p. 134, 2004.

[10] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A*, vol. 70, p. 052328, Nov 2004.

[11] A. Mari and J. Eisert, "Positive wigner functions render classical simulation of quantum computation efficient," *Phys. Rev. Lett.*, vol. 109, p. 230503, Dec 2012.

[12] V. Veitch, N. Wiebe, C. Ferrie, and J. Emerson, "Efficient simulation scheme for a class of quantum optics experiments with non-negative wigner representation," *New Journal of Physics*, vol. 15, no. 1, p. 013037, 2013.

[13] X. Ni, O. Buerschaper, and M. Van den Nest, "A non-commuting stabilizer formalism," *Journal of Mathematical Physics*, vol. 56, no. 5, 2015.

[14] S. D. Bartlett, B. C. Sanders, S. L. Braunstein, and K. Nemoto, "Efficient classical simulation of continuous variable quantum information processes," *Phys. Rev. Lett.*, vol. 88, p. 097904, Feb 2002.

[15] J. Bermejo-Vega, C. Y. Lin, and M. V. den Nest, "The computational power of normalizer circuits over black-box groups," *arXiv:1409.4800*, 2014.

[16] M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 467, no. 2126, pp. 459–472, 2010.

[17] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, (New York, NY, USA), pp. 333–342, ACM, 2011.

[18] M. Sipser, *Introduction to the Theory of Computation, 3rd Ed.* Course Technology, 2012.

[19] S. Aaronson, "Quantum computing, postselection, and probabilistic polynomial-time," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 461, no. 2063, pp. 3473–3482, 2005.

[20] S. Toda, "PP is as hard as the polynomial-time hierarchy," *SIAM J. Comput.,*, vol. 20, no. 5, pp. 865–877, 1991.

Appendix A. Rules for proving results in Table 1

We will show that in Table 1, the theorems labelled by boxed symbols (for example, $\boxed{\mathsf{PH}}$) imply results for all the other cases, and hence for a complete proof of the results in the table, it will suffice to prove just Theorems 1–6 as well as JV 1–6 (save JV1 and JV6). The proof of this is a straightforward consequence of a couple of rules (cf [1]), which we state explicitly here:

- If classical simulation of a set of computational tasks $\mathcal{A}$ is efficient, then classical simulation of any subset of $\mathcal{A}$ would also be efficient.
- If classical simulation of a set of computational tasks $\mathcal{A}$ is hard (#P-hard, QC-hard or PH-collapsing in the sense described above), then classical simulation simulation of any superset of A would also be similarly hard.
- The set of computational tasks with IN(BITS) is a subset of the same set of tasks with IN(PROD). Write this as IN(BITS) $\subset$ IN(PROD). Similarly, OUT(BITS) $\subset$ OUT(PROD), NONADAPT $\subset$ ADAPT.
- If strong simulation of a set of tasks is efficient, then so is the STR(1), STR(n) and WEAK(n) simulation of that set. If any of the latter three notions of simulation is efficient, then WEAK(1)-simulation is efficient (as illustrated in Figure 2). The reverse is also true. For example, if WEAK(1) simulation is #P-hard, then so is WEAK(n)-simulation. As explained in [1], note that #P-hardness holds only for strong notions of simulation, and QC-hardness holds only for weak notions of simulation.

Appendix B. Proof of Theorem 1

A 3-CNF formula f (i.e. a Boolean formula in conjuctive normal form [18]) with n variables and N clauses is of the form

$$f(x_1,\ldots,x_n) = (a_{11}\vee a_{12}\vee a_{13})\wedge(a_{21}\vee a_{22}\vee a_{23})\wedge\ldots\wedge(a_{N1}\vee a_{N2}\vee a_{N3}) \tag{7}$$

where each $a_{ij}\in\{x_1,\ldots,x_n,\overline{x}_1,\ldots,\overline{x}_n\}$. We shall always assume that every variable $x_1,\ldots,x_n$ appears in the formula for f, so that $n\le 3N$, i.e. $n=O(N)$.

We define AbsSAT to be the following problem: Given a 3-CNF formula $f:\{0,1\}^n\to\{0,1\}$, compute

$$S(f) = \left|\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\right|.$$

Note that if we let $\#_i(f) = |\{x|f(x)=i\}|$, then $S(f) = |\#_0(f)-\#_1(f)|$.

Lemma 1. AbsSAT is #P-hard.

Proof. We shall construct a reduction from the #P-complete problem #SAT to AbsSAT. Given a #SAT-instance $\phi(x_1,\ldots,x_n)$, introduce a new variable y and define the Boolean formula

$$\tilde{\phi}(x_1,\ldots,x_n,y) = \phi(x_1,\ldots,x_n)\vee y.$$

Let $A(\varphi)$ denote the set of satisfying assignments to a Boolean formula φ. Then

$$A(\tilde{\phi}) = \{(x_1,\ldots,x_n,0)|(x_1,\ldots,x_n)\in A(\phi)\}\cup\{(x_1,\ldots,x_n,1)|(x_1,\ldots,x_n)\in\{0,1\}^n\}.$$

Hence, $\#_1(\tilde{\phi}) = \#_1(\phi)+2^n$, and $\#_0(\tilde{\phi}) = 2^{n+1}-\#_1(\tilde{\phi}) = 2^n-\#_1(\phi)$. This gives

$$S(\tilde{\phi}) = |\#_0(\tilde{\phi})-\#_1(\tilde{\phi})| = |2^n-\#_1(\phi)-\#_1(\phi)-2^n| = 2\#_1(\phi).$$

Hence, solving the AbsSAT instance $\tilde{\phi}(x_1,\ldots,x_n,y)$ gives $S(\tilde{\phi})$, from which $\#_1(\phi)$ can be found. Therefore, AbsSAT is #P-hard.

□

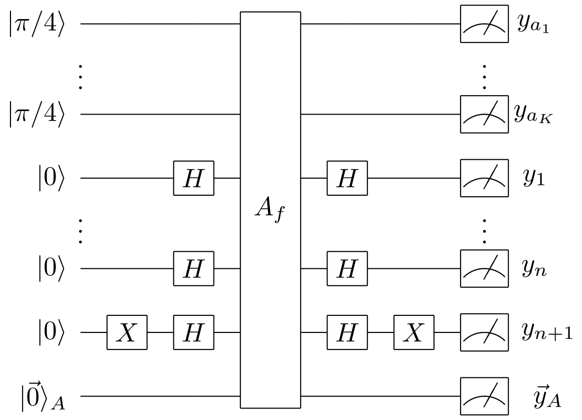
Theorem 1. Let ν = (IN(PROD), NONADAPT, OUT(BITS)). Then the STR(n)-simulation of $\mathcal{C}_\nu$ is #P-hard.

Proof. Assume that there exists an efficient STR(n)-simulation S of $\mathcal{C}_\nu$. We'll use S to construct an efficient algorithm for AbsSAT: On input $f:\{0,1\}^n\to\{0,1\}$, given as a 3-CNF formula with N clauses, where $n=O(N)$, construct a quantum circuit Q_f, consisting of only the basic Clifford gates and T gates, that acts on the following computational basis states as follows: (See Lemma 8 for the details of such a construction)

$$Q_f|x_1,\ldots,x_n,0\rangle|\vec{0}\rangle_A = |x_1,\ldots,x_n,f(x_1,\ldots,x_n)\rangle|\vec{0}\rangle_A.$$

Let K be the number of T gates in Q_f. For the jth T gate (acting on the l_jth line), for $j = 1, \ldots, K$, introduce an ancilla line a_j, and replace the T gate with the CNOT gate CX_{l_j,a_j}. Call the resulting circuit A_f. It is straightforward to check that if each ancilla wire is initialized to the state $|\pi/4\rangle$, and measured at the end of the computation, and if the measurement outcomes are $0 \ldots 0$, then the non-ancilla registers of A_f would implement Q_f. Hence, ignoring the ancilla registers, for the above measurement outcomes, we have $A_f : |x_1, \ldots, x_n, y\rangle|\vec{0}\rangle_A \mapsto |x_1, \ldots, x_n, y \oplus f(x_1, \ldots, x_n)\rangle|\vec{0}\rangle_A$.

Let M_f be the following circuit:



If we postselect the outcomes $y_{a_1} \ldots y_{a_K} = 0 \ldots 0$ for the ancilla registers, the nonancilla registers evolve as follows:

$$\begin{aligned}
|0\ldots0,0\rangle|\vec{0}\rangle_A &\to |0\ldots0,1\rangle|\vec{0}\rangle_A \\
&\to \frac{1}{\sqrt{2^{n+1}}}\sum_x |x\rangle(|0\rangle - |1\rangle)|\vec{0}\rangle_A \\
&\to \frac{1}{\sqrt{2^{n+1}}}\sum_x |x\rangle(|f(x)\rangle - |1 \oplus f(x)\rangle)|\vec{0}\rangle_A \\
&= \frac{1}{\sqrt{2^{n+1}}}\sum_x (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)|\vec{0}\rangle_A \\
&\to \frac{1}{2^n}\sum_{xy} (-1)^{f(x)+x\cdot y}|y\rangle|1\rangle|\vec{0}\rangle_A \\
&\to \frac{1}{2^n}\sum_{xy} (-1)^{f(x)+x\cdot y}|y\rangle|0\rangle|\vec{0}\rangle_A.
\end{aligned}$$

Hence, the conditional probability of obtaining the all-zero string given that the ancilla measurements also reveal the all-zero string is

$$\Pr(0_1 \ldots 0_{n+1}, \vec{0}_A | 0_{a_1} \ldots 0_{a_K}) = \left| \frac{1}{2^n} \sum_x (-1)^{f(x)} \right|^2.$$

But the LHS of the above expression is equal to

$$\Pr(0_1 \ldots 0_{n+1}, \vec{0}_A | 0_{a_1} \ldots 0_{a_K}) = \frac{\Pr(0_{a_1} \ldots 0_{a_K}, 0_1 \ldots 0_{n+1}, \vec{0}_A)}{\Pr(0_{a_1} \ldots 0_{a_K})}.$$

Now, $\Pr(0_{a_1} \ldots 0_{a_K}) = 1/2^K$, since each ancilla bit has a probability of $1/2$ of being measured zero.

Simplifying the above expressions, we get

$$\left| \sum_x (-1)^{f(x)} \right| = 2^{n+K/2} \sqrt{\Pr(0_{a_1} \ldots 0_{a_K}, 0_1 \ldots 0_{n+1}, \vec{0}_A)}.$$

But $\Pr(0_{a_1}\dots 0_{a_K}, 0_1\dots 0_{n+1}, \vec{0}_A)$ is a joint outcome probability, and hence can be obtained by running S on $\langle M_f, 00\dots 0\rangle$. (The input to S is valid since M_f is a non-adaptive Clifford circuit with product state inputs.) Hence, the procedure given is an efficient algorithm for AbsSAT. Since AbsSAT is #P-hard, this implies that $\mathcal{C}_\nu$ is #P-hard as well. □

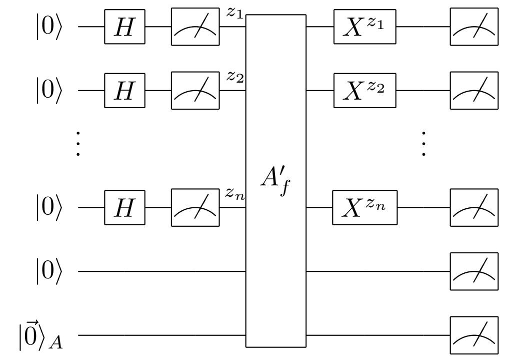
Appendix C. Proof of Theorem 2

Theorem 2. Let $\nu =$ (IN(BITS), ADAPT, OUT(BITS)). Then the $\mathsf{STR}(n)$-simulation of $\mathcal{C}_\nu$ is #P-hard.

Proof. Assume that there exists an efficient $\mathsf{STR}(n)$-simulation S of $\mathcal{C}_\nu$. We'll use S to construct an efficient algorithm M for $\#SAT$, i.e. given as input a 3-CNF formula $f:\{0,1\}^n\to\{0,1\}$, our goal is to find $\#f = \sum_x f(x)$.

M = "On input $f:\{0,1\}^n\to\{0,1\}$, given as a 3-CNF formula,

1. Construct a classical circuit C_f consisting of only Toffoli gates that acts on the following computational basis states as follows: (see Lemma 7 for the details of this construction)
$$C_f(x_1,\dots,x_n,1,\vec{1}_A) = (x_1,\dots,x_n,f(x_1,\dots,x_n),\vec{1}_A).$$
2. Simulate C_f with a Clifford circuit from $\mathcal{A}$: replace each Toffoli gate $T_{abc}(x,y,z)=(x,y,z\oplus xy)$ acting on lines a,b,c with $(CX_{bc})^x M_a(x)$. Call the resulting quantum circuit A_f. The circuit A_f acts on computational basis states as follows:
$$A_f|x_1,\dots,x_n,1\rangle|\vec{1}\rangle_A \to |x_1,\dots,x_n,f(x_1,\dots,x_n)\rangle|\vec{1}\rangle_A.$$
By applying X gates (expressed as $X = HS^2H$) to the appropriate lines at the input and output of A_f, let A'_f be the circuit that acts on computational basis states as follows:
$$A'_f|x_1,\dots,x_n,0\rangle|\vec{0}\rangle_A \to |x_1,\dots,x_n,f(x_1,\dots,x_n)\rangle|\vec{0}\rangle_A.$$
3. Let G_f be the following circuit:



4. Feed $\langle G_f, 00\dots 01\vec{0}_A\rangle$ into S to find $p = p(00\dots 01\vec{0}_A)$, the probability that the output is $00\dots 01\vec{0}_A$.
5. Output $\#f = 2^n p$."

A straightforward calculation shows that the output of G_f on input $|00\dots 0\rangle|\vec{0}\rangle_A$ is $|0,\dots,0,f(z)\rangle|\vec{0}\rangle_A$ if the intermediate measurement results are $z = z_1\dots z_n$ Hence,

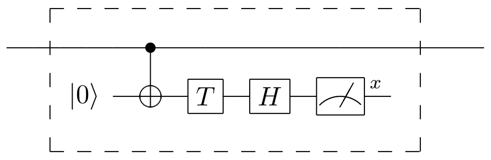
$$\begin{aligned} p = p(0\dots 0,1,\vec{0}_A) &= \sum_z p(0\dots 0,1,\vec{0}_A|z_1\dots z_n)p(z_1\dots z_n) \\ &= \sum_z |\langle 0\dots 01,\vec{0}_A|0\dots 0,f(z),\vec{0}_A\rangle|^2\frac{1}{2^n} \\ &= \frac{1}{2^n}\sum_x f(x). \end{aligned} \tag{8}$$

Hence, the output of M is $2^n p = \#f$.

□

Appendix D. Proof of Theorem 3

We follow a proof similar to that given in [16] that shows that if IQP circuits can be efficiently classically simuated in the weak sense, then the polynomial hierarchy collapses. Recall that the T gate is given by $T = \mathrm{diag}(1, e^{i\pi/4})$. We first consider the following gadget $\mathcal{G}$:



(9)

Lemma 2.

$$\mathcal{G} : |\psi\rangle \mapsto \begin{cases} T|\psi\rangle & \text{if } x = 0 \\ ZT|\psi\rangle & \text{if } x = 1 \end{cases}$$

Proof. Applying the unitary gates in the circuit to the state $|\psi\rangle|0\rangle$ gives $\frac{1}{\sqrt{2}}\left[(T|\psi\rangle)|0\rangle + (ZT|\psi\rangle)|1\rangle\right]$. Hence, we get the desired states when the ancilla wire is measured. □

From the proof of Lemma 2, we note that the measurement outcomes $x = 0, 1$ occur with an equal probability. Note that if $x = 0$, then $\mathcal{G}$ would have implemented the T gate.

Lemma 3. Let Q be an arbitrary quantum circuit comprising the basic Clifford gates and T gates. Let $\nu = $ (IN(BITS), NONADAPT, OUT(PROD)). Then Q with postselection can be weakly simulated by $\mathcal{C}_\nu$ with postselection.

Proof. We first show how we can simulate the circuit Q using circuits from $\mathcal{C}_\nu$ with postselection. For each T gate in Q, we replace it by the gadget $\mathcal{G}$ defined above. If the number of T gates is s, then this procedure produces a new circuit C with s new lines. Now, note that the new circuit C belongs to the class C_ν since the HT gates together with the computational basis measurements implement a product measurement. Now, if we postselect on outcome 0 for all the measurements in the new lines, then each gadget $\mathcal{G}$ would implement the T gate. Hence, $\mathcal{C}_\nu$ with postselection would weakly simulate Q. Now, since we have the resource of postselection, it follows that Q with postselection can be weakly simulated by $\mathcal{C}_\nu$ with postselection. □

We now make the following definition (recall notation in Eq. (6): we use similar notation for conditional probabilities) to capture the power of subsets of Clifford computational tasks with postselection.

Definition 3. ($\mathsf{post}\mathcal{C}_\nu\mathsf{P}$) Let $\mathcal{C}_\nu$ be a subset of Clifford computational tasks. A language $L \in \mathsf{post}\mathcal{C}_\nu\mathsf{P}$ if there exists an error tolerance $0 < \epsilon < \frac{1}{2}$, and a uniform family $\{C_w\}_w$ of circuits in $\mathcal{C}_\nu$ with $n + p(n)$ lines (call these lines $l_1, \ldots, l_n, a_1, \ldots, a_p$, where $p = p(n)$), where $n = |w|$ and p is some polynomial, such that

$$\begin{aligned} p_{C_w}^{\{a_1,\ldots,a_N\}}(00\ldots0) &> 0, \\ w \in L \implies p_{C_w}^{\{l_1\}|\{a_1,\ldots,a_N\}}(1|00\ldots0) &\geq 1-\epsilon, \\ w \notin L \implies p_{C_w}^{\{l_1\}|\{a_1,\ldots,a_N\}}(0|00\ldots0) &\geq 1-\epsilon. \end{aligned} \tag{10}$$

We will use the definition of $\mathsf{postBQP}$ given in [16], which allows for multiple postselected lines. Note that this is equivalent to the definition given in [19] where $\mathsf{postBQP}$ was introduced, which allows for only single lines. We now show that the class just defined is equal to $\mathsf{postBQP}$.

Lemma 4. $\mathsf{post}\mathcal{C}_\nu\mathsf{P} = \mathsf{postBQP}$.

Proof. The forward direction is immediate, since extended Clifford circuits are a special case of general quantum circuits. To prove the backward direction, let $L \in \mathsf{postBQP}$. Then there exists an error tolerance $0 < \epsilon < \frac{1}{2}$, and a uniform family $\{Q_w\}_w$ of quantum circuits consisting of the basic Clifford gates and T gates with $n + p(n)$ lines (call these lines $l_1, \ldots, l_n, b_1, \ldots, b_N$, where $p = p(n)$), where $n = |w|$ and p is some polynomial, such that

$$\begin{aligned} p_{Q_w}^{\{b_1,\ldots,b_N\}}(00\ldots0) &> 0, \\ w \in L \implies p_{Q_w}^{\{l_1\}|\{b_1,\ldots,b_p\}}(1|00\ldots0) &\geq 1-\epsilon, \end{aligned}$$

$$w \notin L \implies p_{Q_w}^{\{l_1\}|\{b_1,\ldots,b_N\}}(0|00\ldots0) \geq 1-\epsilon.$$

By Lemma 3, for each Q_w, there exists an extended Clifford circuit $C_w \in C_\nu$ that, with postselection, simulates Q_w with postselection. If s is the number of T gates in Q_w, then C_w has $n+p(n)+s$ lines. Postselecting on the last $p(n)+s$ lines, it follows that the set of circuits $\{C_w\}$ satisfies the definition given for $\mathsf{post}\mathcal{C}_\nu\mathsf{P}$. Hence, $L \in \mathsf{post}\mathcal{C}_\nu\mathsf{P}$. □

Lemma 5. Let $\nu =$ (IN(BITS), NONADAPT, OUT(PROD)). If $\mathcal{C}_\nu \in \mathsf{PWEAK}(n)$, then $\mathsf{post}\mathcal{C}_\nu\mathsf{P} \subseteq \mathsf{postBPP}$.

Proof. Let $L \in \mathsf{post}\mathcal{C}_\nu\mathsf{P}$. Then there exists an error tolerance $0 < \epsilon < \frac{1}{2}$, and a uniform family $\{C_w\}_w$ of circuits in $\mathcal{C}_\nu$ with $n+p(n)$ lines (call these lines $l_1,\ldots,l_n,a_1,\ldots,a_p$, where $p=p(n)$), where $n=|w|$ and p is some polynomial, such that Eq. (10) holds.

But $\mathcal{C}_\nu \in \mathsf{PWEAK}(n)$. Hence, for all circuits $Q_w \in \mathcal{C}_\nu$, there exists a classical randomized circuit C_w with $n+p$ lines such that

$$p_{Q_w}^{\{l_1,\ldots,l_n,a_1,\ldots,a_p\}}(y) = p_{C_w}^{\{l_1,\ldots,l_n,a_1,\ldots,a_p\}}(y).$$

For any subsets $I, J \subseteq [n]$ of lines, similar relations hold for marginal probabilities and conditional probabilities: $p_{Q_w}^{I}(y) = p_{C_w}^{I}(y)$ and $p_{Q_w}^{I|J}(y|z) = p_{C_w}^{I|J}(y|z)$. This implies that

$$p_{Q_w}^{\{l_1\}|\{a_1,\ldots,a_p\}}(1|00\ldots0) = p_{C_w}^{\{l_1\}|\{a_1,\ldots,a_p\}}(1|00\ldots0),$$

and hence Q_w obey Eq. (10). This implies that $L \in \mathsf{postBPP}$. Therefore, $\mathsf{post}\mathcal{C}_\nu\mathsf{P} \subseteq \mathsf{postBPP}$. □

Theorem 3. Let $\nu =$ (IN(BITS), NONADAPT, OUT(PROD)). If $\mathcal{C}_\nu \in \mathsf{PWEAK}(n)$, then PH collapses to the third level.

Proof. By Lemmas 4 and 5, if $\mathcal{C}_\nu \in \mathsf{PWEAK}(n)$, then $\mathcal{C}_\nu \in \mathsf{PWEAK}(n)$, then $\mathsf{postBPP} \supseteq \mathsf{post}\mathcal{C}_\nu\mathsf{P} = \mathsf{postBQP}$. By Aaronson's Theorem, $\mathsf{PP} = \mathsf{postBQP}$ [19], and by Toda's Theorem $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$ [20]. Hence, we get the following string of inclusions:

$$\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\mathsf{postBQP}} = \mathsf{P}^{\mathsf{post}\mathcal{C}_\nu\mathsf{P}} \subseteq \mathsf{P}^{\mathsf{postBPP}} \subseteq \mathsf{P}^{\mathsf{BPP}^{\mathsf{NP}}} = \mathsf{BPP}^{\mathsf{NP}} \subseteq \Sigma_3^p,$$

which implies that PH collapses to the third level. □

Appendix E. Proof of Theorem 4

Consider the proof of Theorem 1. Note that the circuit M_f is unitary. Hence, an even stronger result than Theorem 1 is true: if we replaced nonadaptive circuits with unitary ones (call this UNITARY), the simulation complexity is still #P-hard. In other words,

Lemma 6. Let $\nu =$ (IN(PROD), UNITARY, OUT(BITS)). Then the $\mathsf{STR}(n)$-simulation of $\mathcal{C}_\nu$ is #P-hard.

The $\mathsf{STR}(n)$-simulation of $\mathcal{C}_\nu$ is equivalent to the following problem:

Input: $\langle T, y\rangle$, where $T = (|x\rangle, B, |\alpha\rangle)$, B is a unitary circuit, $x \in \{0,1\}^n$, $\alpha = \alpha_1\ldots\alpha_n$ and each $|\alpha_i\rangle \in \mathbb{C}_2$.
Output: $p_T(y) = |\langle \alpha_1^{y_1}\ldots\alpha_n^{y_n}|B|x\rangle|^2$.

Now, let $\mu =$ (IN(BITS), UNITARY, OUT(PROD)), then the $\mathsf{STR}(n)$-simulation of $\mathcal{C}_\mu$ is equivalent to the following problem:

Input: $\langle T', y\rangle$, where $T' = (|\alpha_1^{y_1}\ldots\alpha_n^{y_n}\rangle, B^\dagger, \{I\}_i)$, B is a unitary circuit
Output: $p_{T'}(x) = |\langle x|B^\dagger|\alpha_1^{y_1}\ldots\alpha_n^{y_n}\rangle|^2 = |\langle\alpha_1^{y_1}\ldots\alpha_n^{y_n}|B|x\rangle|^2 = p_T(y)$.

Since both problem instances can be transformed easily to each other, and since both problems involve calculating the same quantity, we conclude that the $\mathsf{STR}(n)$-simulation of $\mathcal{C}_\mu$ is also #P-hard. If it is #P-hard to simulate this class of unitary circuits, then it must be #P-hard to simulate the same class but with unitary circuits replaced by nonadaptive circuits. Therefore, we obtain the following theorem:

Theorem 4. Let $\nu =$ (IN(BITS), NONADAPT, OUT(PROD)). Then the $\mathsf{STR}(n)$-simulation of $\mathcal{C}_\nu$ is #P-hard.

Appendix F. Proof of Theorem 5

Theorem 5. Let $\nu =$ (IN(PROD), NONADAPT, OUT(PROD)). Then $\mathcal{C}_\nu \in \mathsf{PSTR}(1)$.

Proof. We use the following notation: for any single-qubit operator O, let $O_1 = O \otimes I \otimes \ldots \otimes I$. Given a Clifford computational task $T = (|\alpha_1 \ldots \alpha_n\rangle, B, \{U, I, \ldots, I\}) \in \mathcal{C}_\nu$, and a bit $i \in \{0,1\}$, we shall describe an algorithm to compute $p_i := p_T^{\{1\}}(i)$. WLOG, B is a unitary circuit.

Since $p_0 + p_1 = 1$, it suffices to be able to calculate $p_0 - p_1$ efficiently. By Born's rule, this is given by

$$p_0 - p_1 = \langle \alpha | B^\dagger (UZU^\dagger)_1 B | \alpha \rangle. \tag{11}$$

Since the Pauli matrices $\{\sigma^i\}_i$ form a basis for the set of 2×2 matrices , we can write

$$U = \sum_{i=0}^{3} a_i \sigma^i,$$

for some $a_i \in \mathbb{C}$. Hence,

$$UZU^\dagger = \sum_{ij} a_i \overline{a}_j \sigma^i Z \sigma^j.$$

But $\sigma^i Z \sigma^j$ is a Pauli operator. Since the basic Clifford gates map Pauli operators to Pauli operators,

$$B^\dagger (\sigma^i Z \sigma^j)_1 B = \gamma_{ij} P_1^{ij} \otimes \ldots \otimes P_n^{ij}.$$

Putting this into Eq. (11), we get an expression for $p_0 - p_1$.

$$\begin{aligned} p_0 - p_1 &= \sum_{i,j=0}^{3} a_i \overline{a}_j \gamma_{ij} \langle \alpha_1 \ldots \alpha_n | P_1^{ij} \otimes \ldots \otimes P_n^{ij} | \alpha_1 \ldots \alpha_n \rangle \\ &= \sum_{i,j=0}^{3} a_i \overline{a}_j \gamma_{ij} \prod_{k=1}^{n} \langle \alpha_k | P_k^{ij} | \alpha_k \rangle. \end{aligned} \tag{12}$$

We now analyze the running time of our algorithm. Computing $\gamma_{ij} P_1^{ij} \otimes \ldots \otimes P_n^{ij}$ takes $O(n^2)$-time. The formula given in Eq. (12) involves a sum of 9 terms. Each term involves computing n expectation values of 2×2 matrices. Hence, this step takes $O(n)$-time. Overall, the algorithm runs in $O(n^2) = O(N^2)$-time, where N is the number of gates in the circuit (which we assumed to contain no extraneous lines). Hence, $\mathcal{C}_\nu \in \mathsf{PSTR}(1)$.

□

Appendix G. Proof of Theorem 6

Theorem 6. Let $\nu =$ (IN(BITS), ADAPT, OUT(PROD)). Then $\mathcal{C}_\nu \in \mathsf{PWEAK}(1)$.

Proof. This is a special case of the results in Section VIIC of [10], which showed that an IN(BITS), NONADAPT, OUT(BITS) circuit containing d non-Clifford gates, where each gate acts on at most b qubits, can be classically simulated in the WEAK(1) sense in $O(4^{2bd} n + n^2)$-time. In our case, the circuits in C_ν can be thought of as containing exactly one non-Clifford gate on the first wire just before the computational-basis measurement. Hence, $d = b = 1$, which implies that the algorithm runs in $O(n^2)$-time. This concludes the proof that $\mathcal{C}_\nu \in \mathsf{PWEAK}(1)$.

□

Appendix H. Constructing circuits for 3-CNF formulas

In the proof of Theorem 1, we used the fact that given a 3-CNF formula $f : \{0,1\}^n \to \{0,1\}$, we can efficiently construct a quantum circuit A_f comprising only the basic Clifford operations and T gates, which acts on the following computational basis states as follows:

$$A_f |x\rangle |0\rangle |0\rangle_A = |x\rangle |f(x)\rangle |0\rangle_A, \tag{13}$$

where $x \in \{0,1\}^n$, $y \in \{0,1\}$ and $|\cdot\rangle_A$ is an ancilla register of size $O(n)$.

A similar fact was used in the proof of Theorem 2, namely that given a 3-CNF formula $f : \{0,1\}^n \to \{0,1\}$, we can efficiently construct a classical circuit C_f comprising only Toffoli gates, which acts on the following computational basis states as follows:

$$C_f(x, 1, 1_A) = (x, f(x), 1_A), \tag{14}$$

where $x \in \{0,1\}^n$, $y \in \{0,1\}$ and A is an ancilla register of size $O(n)$.

Note that in both circuits C_f and A_f, we do not allow for the addition of more ancilla lines or for the discarding of any bit or qubits. This is because for the notion of $\mathsf{STR}(n)$ simulation, all bit or qubit lines have to be accounted for. Hence, we make explicit the reference to the ancilla registers A. In this section, we present the details of the above constructions.

Recall the definition of a 3-CNF formula given in Eq. (7). As above, we assume that every variable $x_1, \dots, x_n$ appears in the formula for f, so that $n \le 3N$, i.e. $n = O(N)$.

H.1. **Constructing C_f.** We show that we can implement the function f using Toffoli gates alone. We denote the action of the Toffoli gate on lines i, j, k with inputs a, b, c by

$$\mathrm{Tof}_{ijk}(\dots, a, \dots, b, \dots, c, \dots) = (\dots, a, \dots, b, \dots, c \oplus a \cdot b, \dots).$$

We use subscripts at the end to indicate a 'marginalizing out' of the values of all other wires, for example,

$$\mathrm{Tof}_{143}(a, b, c, d, e)_{235} = (a, b, c \oplus a \cdot d, d, e)_{235} = (b, c \oplus a \cdot d, e).$$

Lemma 7. Let f be a 3-CNF formula of the form given by Eq. (7) with n variables and N clauses, where $n = O(N)$. Then there exists a classical circuit C_f consisting of $O(N)$ Toffoli gates on $n + 1 + s(N)$ lines, for some $s(N) = O(N)$ (where we do not allow for the addition of bit lines or the discarding of any bits), such that

$$C_f(x_1, \dots, x_n, 1, \underbrace{1, \dots, 1}_{s(N)}) = (x_1, \dots, x_n, f(x_1, \dots, x_n), \underbrace{1, \dots, 1}_{s(N)}). \tag{15}$$

Remark. The ancilla bits are initialized to 1 instead of 0. This is because the Toffoli gate is universal only if we have the ability to prepare the state 1. In particular, if the inputs were always just 0's, then it would not be possible to create the state 1. On the other hand, we can prepare 0 from 1 since the target bit of Tof(1, 1, 1) is 0.

Proof. We first show how to compute f using AND, OR, NOT, COPY and SWAP gates on the input $(x_1, \dots, x_n)$. Let k_i and $\overline{k}_i$ be the number of times x_i and $\overline{x}_i$, respectively, appear as literals in the formula for f, i.e. $\sum_i (k_i + \overline{k}_i) = 3N$. By assumption, every variable $x_1, \dots, x_n$ appears in the formula for f, so $k_i + \overline{k}_i > 0$ for all i.

For each i, if $k_i > 0$, apply the COPY gate $k_i - 1$ times to x_i and the COPY gate followed by the NOT gate $\overline{k}_i$ times to x_i. Otherwise, if $k_i = 0$ (i.e. $\overline{k}_i > 0$), apply the NOT gate followed by the COPY gate $\overline{k}_i$ times to x_i. This creates the state

$$(\underbrace{x_1, \dots, x_1}_{k_1}, \dots, \underbrace{x_n, \dots, x_n}_{k_n}, \underbrace{\overline{x}_1, \dots, \overline{x}_1}_{\overline{k}_1}, \dots, \underbrace{\overline{x}_n, \dots, \overline{x}_n}_{\overline{k}_n}).$$

Note that the number of gates that the above procedure involves is $\sum_{k_i > 0} \left[(k_i - 1) + 2\overline{k}_i\right] + \sum_{k_i = 0} 2\overline{k}_i \le 2\sum_i (k_i + \overline{k}_i) = 6N$.

Applying the SWAP gate up to $3N$ times to the above state, we get the state

$$(a_{11}, a_{12}, a_{13}, a_{21}, \dots, a_{N1}, a_{N2}, a_{N3}). \tag{16}$$

We now apply the OR and AND gates according to the formula in Eq. (7) to get $(a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \dots \wedge (a_{N1} \vee a_{N2} \vee a_{N3})$. This involves a total of $2N$ OR gates and $N - 1$ AND gates. Hence, the resulting circuit B_f, whose number of gates is bounded above by $6N + 3N + 2N + N - 1 = O(N)$, computes:

$$B_f(x_1, \dots, x_n) = f(x_1, \dots, x_n).$$

Note that the maximum width of B_f, which occurs when the state is given by (16), is $3N$.

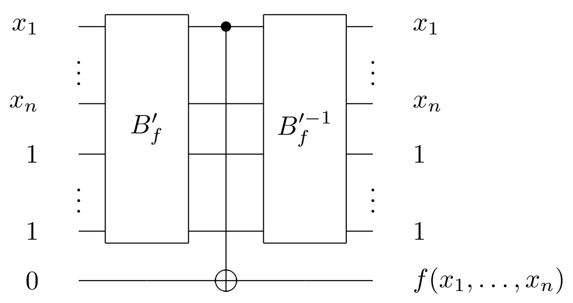


FIGURE 3. Uncomputation trick, in which the output bits, except for those in the target register, are reset to their input values. The state evolves as follows: $(x_1,\dots x_n,1,\dots,1,0) \to (f(x_1,\dots,x_n),j_2,\dots,j_{a(N)},0) \to (f(x_1,\dots,x_n),j_2,\dots,j_{a(N)},f(x_1,\dots,x_n)) \to (x_1,\dots x_n,1,\dots,1,f(x_1,\dots,x_n))$.

We now use the fact that the Toffoli gate together with the ability to prepare the ancilla state 1 is universal for classical computing. In particular, they simulate the above gates as follows:

$$\begin{aligned}
\neg x &= \mathrm{Tof}_{123}(1,1,x)_3,\\
x \wedge y &= [\mathrm{Tof}_{123}\circ \mathrm{Tof}_{453}(x,y,1,1,1)]_3,\\
\mathrm{COPY}(x) &= [\mathrm{Tof}_{123}\circ \mathrm{Tof}_{243}(x,1,1,1)]_{13},\\
x \vee y &= [\mathrm{Tof}_{453}\circ \mathrm{Tof}_{123}\circ \mathrm{Tof}_{453}\circ \mathrm{Tof}_{342}\circ \mathrm{Tof}_{341}(x,y,1,1,1)]_3,\\
\mathrm{SWAP}(x,y) &= [\mathrm{Tof}_{123}\circ \mathrm{Tof}_{321}\circ \mathrm{Tof}_{123}(x,1,y)]_{13}.
\end{aligned} \tag{17}$$

We append ancilla lines initialized to 1 to B_f, and replace all the gates in B_f by Toffoli gates according to the rules in Eq. (17), and apply additional swap gates (implemented by Toffoli gates) so that the first output of the circuit is $f(x_1,\dots,x_n)$. Note that we do not discard any bits. Each of the replacements increases the number of ancilla lines by at most 3 and the number of gates by at most 4. Hence, both the total number of lines $a(N)$ and the number of Toffoli gates in the new circuit B'_f are still $O(N)$. The action of B'_f on the computational basis states is given by:

$$B'_f(x_1,\dots,x_n,\vec{1}) = (f(x_1,\dots,x_n),j_2,\dots,j_{a(N)}),$$

where $(j_2,\dots,j_{a(N)})$ are junk bits.

We now make use of the *uncomputation trick* to reset the junk bits to 1. Since the Toffoli gates are their own inverse, the inverse of B'_f is obtained by applying the gates in B'_f in the reverse order. Consider the circuit B''_f that is formed as follows: first apply B'_f to $(x_1,\dots,x_n,\vec{1})$. Introduce a new ancilla line, called a, initialized to 0. Next, apply the CNOT gate CX_{1a}. Finally, apply B'^{-1}_f to the first $a(N)$ bits to reset them back to $(x_1,\dots,x_n,\vec{1})$. A circuit diagram for the above steps is shown in Figure 3. This gives

$$B''_f(x_1,\dots,x_n,\vec{1},0) = (x_1,\dots,x_n,\vec{1},f(x_1,\dots,x_n)).$$

To get the required circuit C_f, we need to perform three more simple steps. First, the ancilla bit in the last register has to start from 1 instead of 0. This can be achieved by applying a NOT gate (implemented by the Toffoli gate and ancillas initialized to 1) to 0. Second, the CNOT gate has to be simulated by a Toffoli gate. This may be achieved by using the fact that

$$CX(a,b)_{12} = \mathrm{Tof}_{123}(a,1,b)_{13}.$$

Third, the output has to be of the form (15). This is obtained by applying swap gates at the end of the circuit. These steps add at most a constant number of gates and a constant number of ancilla bits. Hence, the resulting circuit C_f has $O(N)$ gates acting on $O(N)$ lines.

□

H.2. **Constructing Q_f.** We now show how we can convert C_f to a circuit Q_f that involves only the basic Clifford gates and the T gate.

Lemma 8. Let f be a 3-CNF formula of the form given by Eq. (7) with n variables and N clauses, where $n = O(N)$. Then there exists a quantum circuit Q_f consisting of $O(N)$ basic Clifford gates and T gates on $n + 1 + s(N)$ lines (where we do not allow for the addition of qubit lines or the discarding of any qubits), such that

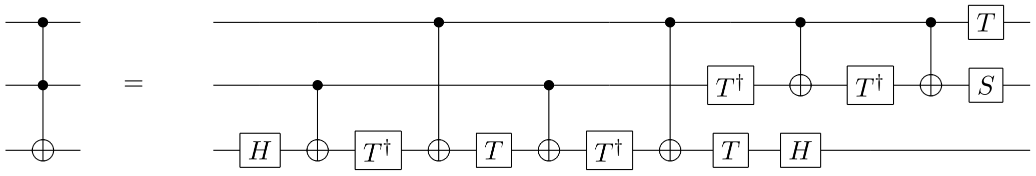
$$Q_f \left| x_1, \ldots, x_n, 0, 0^{s(N)} \right\rangle = \left| x_1, \ldots, x_n, f(x_1, \ldots, x_n), 0^{s(N)} \right\rangle, \tag{18}$$

for some $s(N) = O(N)$.

Proof. Using Lemma 7, we have a circuit C_f comprising $O(N)$ Toffoli gates satisfying

$$C_f \left(x_1, \ldots, x_n, 1, 1^{s(N)} \right) = \left(x_1, \ldots, x_n, f(x_1, \ldots, x_n), 1^{s(N)} \right).$$

Using the construction presented in [8], we express each Toffoli gate in terms of the basic Clifford gates, T and $T^\dagger$ gates, as follows:



Since $T^8 = 1$, we replace each $T^\dagger$ gate above by T^7. These replacements increase the number of gates by a constant factor, and hence the total number of gates in the new circuit is still $O(N)$. Finally, we insert X (expressed as X $= HS^2H$) gates at the start and end of the circuit so that the ancilla lines start and terminate in the state $|0\rangle$. This gives us a quantum circuit obeying Eq. (18) with $O(N)$ wires and $O(N)$ gates.

□