Towards Large-scale Functional Verification of Universal Quantum Circuits

Matthew Amy
Institute for Quantum Computing and David R. Cheriton School of Computer Science
University of Waterloo, Canada
meamy@uwaterloo.ca

We introduce a framework for the formal specification and verification of quantum circuits based on the Feynman path integral. Our formalism, built around exponential sums of polynomial functions, provides a structured and natural way of specifying quantum operations, particularly for quantum implementations of classical functions. Verification of circuits over all levels of the Clifford hierarchy with respect to either a specification or reference circuit is enabled by a novel rewrite system for exponential sums with free variables. Our algorithm is further shown to give a polynomial-time decision procedure for checking the equivalence of Clifford group circuits. We evaluate our methods by performing automated verification of optimized Clifford+T circuits with up to 100 qubits and thousands of T gates, as well as the functional verification of quantum algorithms using hundreds of qubits. Our experiments culminate in the automated verification of the Hidden Shift algorithm for a class of Boolean functions in a fraction of the time it has taken recent algorithms to simulate.

1 Introduction

Verification is a fundamental aspect of modern electronic design. Without a high level of assurance that a circuit design conforms to a particular specification, chip makers stand to lose hundreds of millions of dollars when their product is inevitably recalled. The consequences in the quantum computing realm aren't quite as clear, as the largely software-like nature of quantum circuits alleviates much of the risk associated with design flaws. On the other hand, quantum resource analyses, which typically vary wildly between compilers [19], are currently being used to assess and guide real security policies [3, 18], so it is highly desirable to attain some degree of assurance that these resource analyses are indeed correct.

Due to the absence of large, universal quantum computers and the inherent difficulty of simulating quantum circuits, testing is generally not a viable option for verification. By contrast, various methods of formal verification have been developed for quantum circuits and programs, including equivalence checking [7, 30, 31], diagrammatic methods [13, 15], model checkers [6, 16], program logics [32] and formal proof [27]. However, two questions remain: how can the intended effect of a quantum program be specified in a clear, human readable and verifiable way, and how can we scale automated verification to large circuits?

Typical *functional* verification methods – verification of the precise input-output relation – either verify equivalence against a simpler circuit or diagrammatic implementation (e.g., [15, 30, 31]), or a matrix representation such as a unitary or superoperator (e.g., [27]). With either approach, errors can creep in *on the specification side*, as both circuit and matrix presentations can be difficult for humans to write and understand. Moreover, in the former case it is assumed that a certified implementation exists in the first place, and in the latter case the matrix either

Preliminary Report. Final version to appear in:
QPL 2018

requires exponential space to write and store, or is left abstract [27], relying on structural proofs which are generally not suitable for verifying heavily optimized circuits.

In this work we propose a novel framework for the formal specification and functional verification of unitary (i.e., measurement-free) quantum circuits over a universal gate set – specifically, the Clifford group extended with Z-axis rotations taken from the Clifford hierarchy [17]. Our framework is built around Richard Feynman's *path integral* technique, which has been used recently to prove results in complexity theory [12,24], and to perform circuit simulation [10,21] and optimization [1,2,4]. Specifically, we develop a concrete representation of quantum operators as *path-sums* – exponential sums of basis states over a finite set of Boolean *path variables.* Our path-sums directly coincide with the standard mathematical presentation of common quantum circuits and algorithms (e.g., [25]), and further allow the direct use of classical functions, which can themselves be tested or otherwise verified, to formally specify quantum operations.

To verify quantum circuits, we give a computable, compositional semantics of quantum circuits as path-sums. We show that over Clifford+R_k circuits for any fixed k, this interpretation is efficiently computable and compact. We then present a reduction system for path-sums which iteratively reduces the number of path variables until a (non-unique) normal form is reached. Our reduction system together with an efficient initial transformation is complete for Clifford group circuits, giving a polynomial-time equivalence checking algorithm. Experimentally, we use our reduction system to perform the automated verification of optimized Clifford+T circuits, as well as Clifford+R_k implementations of various quantum algorithms against formal specifications as path-sums for up to 200 qubits.

Preliminaries We work in the strictly unitary picture of quantum computing [25] – that is, quantum computations are modelled by unitary operators on a complex vector space of dimension 2^n. While we do not consider measurements, we allow qubit initialization, corresponding to partial isometries on a complex vector space. We denote the computational basis vectors as $|\mathbf{x}\rangle$ for binary strings $\mathbf{x} = x_1x_2\dots x_n \in \mathbb{Z}_2^n$.

A circuit is defined as a sequence of quantum gates applied to individual qubits. We primarily consider three quantum gates:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad R_k = \begin{pmatrix}1 & 0\\ 0 & e^{\frac{2\pi i}{2^k}}\end{pmatrix}, \quad \text{and} \quad \mathrm{CNOT} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

For $k \geq 1$, all three gates lie in the kth level of the *Clifford hierarchy*, denoted $\mathcal{C}_k$, where $\mathcal{C}_k = \{U | U\mathcal{C}_1U^\dagger \subseteq \mathcal{C}_{k-1}\}$ and $\mathcal{C}_1$ is the Pauli group. Two important cases are the *Clifford group* ($\mathcal{C}_2$) and *Clifford+T* ($\mathcal{C}_3$). While for $k \leq 3$ the above gates suffice to generate $\mathcal{C}_k$, it is not generally known whether $\mathcal{C}_k = \langle H, R_k, \mathrm{CNOT}\rangle$.

Much of our formalism involves polynomial representations of pseudo-Boolean functions – functions from $\mathbb{Z}_2^n$ into some set S. In particular, we are interested in pseudo-Boolean functions into the ring of *dyadic fractions* $\mathbb{D} = \{\frac{a}{2^b} | a, b \in \mathbb{Z}\}$, which correspond uniquely to multilinear polynomials in $\mathbb{D}_M[\mathbf{x}] = \mathbb{D}[\mathbf{x}]/\langle x_i^2 - x_i\rangle$. In our context the ring of dyadic fractions arises from the phase factors of R_k gates, and are needed to precisely represent the quantum Fourier transform.

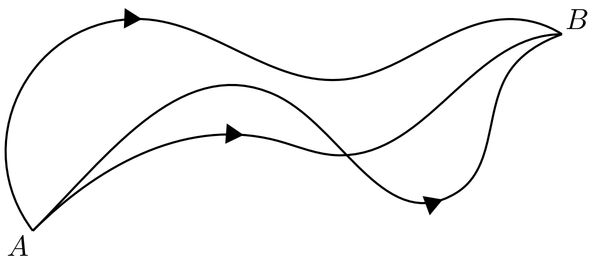


Figure 1: The paths of a particle from point A to B.

2 The path-sum framework

The path-sum dates back to Feynman and the path integral formulation of quantum mechanics [14]. In a general sense, the idea is to describe the amplitude of a particular state (say, of a particle) by an integral over all possible paths leading to that state. Figure 1 shows the trajectories of a particle moving from states A to B – in the path integral formulation, the final amplitude is described as the sum of the amplitudes of each path. The output amplitudes of a quantum circuit, as a quantum mechanical system, can likewise be described as the sum over all trajectories of the system. However, as quantum gates are typically modelled as operators on a finite dimensional Hilbert space, a discrete sum rather than integral is typically used [8, 12, 21, 24].

We can describe a path-sum abstractly as a discrete set of *paths* $S \subseteq \mathbb{Z}_2^m$, together with an amplitude function ϕ and state transformation f representing the operator

$$U : |\mathbf{x}\rangle \mapsto \sum_{\mathbf{y}\in S} \phi(\mathbf{x},\mathbf{y})|f(\mathbf{x},\mathbf{y})\rangle.$$

In this form, the path-sum is not particularly useful as a computational representation, as the representations of ϕ and f are not fixed – indeed ϕ itself may be a unitary matrix with $\phi(\mathbf{x},\mathbf{y})$ indexing a particular entry. Instead, we fix a concrete representation based on multivariate polynomials which suffices to exactly represent most interesting quantum operations.

Definition 2.1 (path-sum)**.** An n-qubit *path-sum* ξ consists of

- an *input signature* $|\mathbf{x} = x_1x_2\cdots x_n\rangle$ where each x_i is a (distinct) variable or Boolean constant,
- a *phase polynomial* $P \in \mathbb{D}_M[\mathbf{x},\mathbf{y}]$ over input variables $\mathbf{x}$ and *path variables* $\mathbf{y} = y_1y_2\dots y_m$, and
- an *output signature* $|f(\mathbf{x},\mathbf{y}) = f_1(\mathbf{x},\mathbf{y})\cdots f_n(\mathbf{x},\mathbf{y})\rangle$ where each $f_i \in \mathbb{Z}_2[\mathbf{x},\mathbf{y}]$ is a Boolean polynomial.

The *associated operator* of a path-sum is the partial linear map U_ξ where

$$U_\xi : |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^m}} \sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle.$$

We say a path variable is *internal* if it does not appear in the output signature. Our presentation is inspired by descriptions of quantum operators in mathematical texts [20, 25], and as such we write a path-sum informally by the action of its associated operator. By an abuse of notation, we use $|\mathbf{x}\rangle$ to refer to either an input signature or an arbitrary Boolean vector corresponding to an input signature.

Example 2.2. Path-sum representations of common quantum gates and circuits are listed below:

$$\begin{aligned}
T :&|x\rangle \mapsto e^{2\pi i \frac{x}{8}}|x\rangle \\
H :&|x\rangle \mapsto \frac{1}{\sqrt{2}}\sum_{y\in\mathbb{Z}_2} e^{2\pi i \frac{xy}{2}}|y\rangle \\
\mathsf{Toffoli}_n :&|x_1x_2\cdots x_n\rangle \mapsto |x_1x_2\cdots(x_n \oplus \textstyle\prod_{i=1}^{n-1} x_i)\rangle \\
\mathsf{Adder}_n :&|\mathbf{x}\rangle|\mathbf{y}\rangle|\mathbf{0}\rangle \mapsto |\mathbf{x}\rangle|\mathbf{y}\rangle|\mathbf{x}+\mathbf{y}\rangle \\
\mathsf{QFT}_n :&|\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^n}}\sum_{\mathbf{y}\in\mathbb{Z}_2^n} e^{2\pi i \frac{[\mathbf{x}\cdot\mathbf{y}]}{2^n}}|\mathbf{y}\rangle
\end{aligned}$$

Addition and multiplication of Boolean vectors are interpreted as integer operations at the bit level. In the QFT above, $[\mathbf{x}\cdot\mathbf{y}]$ denotes the integer value of $\mathbf{x}\cdot\mathbf{y}$. For any classical function f, we can lift the polynomial representation of f to a quantum operator via the path-sum $|\mathbf{x}\rangle|\mathbf{0}\rangle \mapsto |\mathbf{x}\rangle|f(\mathbf{x})\rangle$. Note that the polynomial representation of a classical function may grow exponentially large, as in the case of addition. A practical implementation of path-sums as a specification language would include a classical sub-language, along with a verified translation from such programs into Boolean polynomials.

As a unitary or partial isometry may admit many distinct path-sum representations, we define an equivalence between path-sums with the same associated operator.

Definition 2.3 (equivalence)**.** Two path-sums ξ_1, ξ_2 are *equivalent*, denoted $\xi_1 \equiv \xi_2$, if and only if their associated operators are equal – that is, $U_{\xi_1} = U_{\xi_2}$.

An additional point to note is that non-isometric path-sums are possible in our framework, as for instance $|x\rangle \mapsto |0\rangle$ is a valid path-sum. In this work we are concerned only with the unitary circuit model and by extension isometric path-sums, hence we define a notion of well-formedness for path-sums.

Definition 2.4 (well-formed)**.** A path-sum is well-formed if its associated operator is a (partial) isometry.

In practice, well-formedness is only an issue when writing path-sums directly as specifications, and our verification methods work even when a path-sum is not guaranteed to be well-formed. We leave it as a question for future research to determine methods for checking well-formedness of path-sums.

2.1 Compositions of path-sums

As with quantum circuits, path-sums may be composed both *vertically* and *horizontally* – that is, composed in parallel with another path-sum on a distinct subsystem or in sequence on the same subsystem, respectively. Vertical composition is defined in the obvious way – concatenating the inputs and outputs then adding the phase polynomials with appropriate renaming – but horizontal composition requires more care.

Intuitively, as path-sums symbolically describe mappings between linear combinations of basis vectors, we can compose the output $|f(\mathbf{x},\mathbf{y})\rangle$ of one path-sum with the input $|\mathbf{x}'\rangle$ of another by substituting each input value x_i' with the corresponding output $f_i(\mathbf{x},\mathbf{y})$. For instance, we can compute the composition of $|x_1x_2x_3\rangle \mapsto |x_1(x_1\oplus x_2)x_3\rangle$ followed by $|x_1'x_2'x_3'\rangle \mapsto |x_1'x_2'(x_2'\oplus x_3')\rangle$ by substituting x_2' with $x_1\oplus x_2$:

$$|x_1x_2x_3\rangle \mapsto |x_1(x_1\oplus x_2)(x_1\oplus x_2\oplus x_3)\rangle.$$

However, this presents a problem when the path-sum on the left (i.e. right-to-left composition) is a partial isometry, as we may end up composing a variable $f_i(\mathbf{x},\mathbf{y}) = x_j$ with a constant state $x_i' = b$ for some $b \in \mathbb{Z}_2$, effectively post-selecting on $x_j = b$. For this reason we require that only *compatible*[1] signatures are composed; in particular, an output $|f(\mathbf{x},\mathbf{y})\rangle$ is compatible with an input $|\mathbf{x}'\rangle$ if and only if for every i, either x_i' is a variable or $x_i' = b = f_i(\mathbf{x},\mathbf{y})$.

When the left-most path-sum has a non-zero phase polynomial, substitutions may extend to the phase. As the phase and output polynomials are defined over different rings ($\mathbb{D}$ and $\mathbb{Z}_2$, respectively), when substituting a variable with a Boolean polynomial in the phase we first need to lift it into a *functionally equivalent* polynomial over $\mathbb{D}$. For instance, for all $x, y \in \mathbb{Z}_2$, $\frac{1}{4}(x\oplus y) = \frac{1}{4}x + \frac{1}{4}y - \frac{1}{2}xy$. We define the *lifting* of a Boolean polynomial P to a polynomial $\overline{P} \in \mathbb{D}_M[\mathbf{x}]$ recursively by

$$\overline{\mathbf{x}^\alpha} = \mathbf{x}^\alpha,$$
$$\overline{P+Q} = \overline{P} + \overline{Q} - 2\overline{PQ},$$

where $\mathbf{x}^\alpha = x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ for $\alpha \in \mathbb{Z}_2^n$ is a multi-index, and the first equation uses the natural inclusion of $\mathbb{Z}_2$ in $\mathbb{D}$. It can be easily verified that the lifting of a Boolean polynomial preserves its action on elements of $\mathbb{Z}_2$.

Lemma 2.5. *For any Boolean-valued polynomial P and all $\mathbf{x} \in \mathbb{Z}_2^n$, $\overline{P}(\mathbf{x}) = P(\mathbf{x}) \mod 2$.*

We can now formally define the functional composition of path-sums.

Definition 2.6. (sequential composition)
The *sequential composition* of two compatible path-sums

$$U_\xi : |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^m}} \sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle, \qquad U_{\xi'} : |\mathbf{x}'\rangle \mapsto \frac{1}{\sqrt{2^{m'}}} \sum_{\mathbf{y}'\in\mathbb{Z}_2^{m'}} e^{2\pi i P'(\mathbf{x},\mathbf{y})}|f'(\mathbf{x}',\mathbf{y}')\rangle,$$

denoted $\xi' \circ \xi$, is given by

$$U_{\xi'\circ\xi} : |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^{m+m'}}} \sum_{\mathbf{y}\in\mathbb{Z}_2^{m+m'}} e^{2\pi i\left(P + P'[y_i\leftarrow y_{i+m}][x_i'\leftarrow \overline{f_i}]\right)(\mathbf{x},\mathbf{y})}|\left(f'[x_i' \leftarrow f_i]\right)(\mathbf{x},\mathbf{y})\rangle,$$

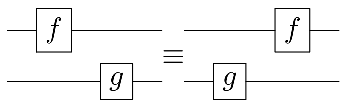
where $P[x \leftarrow Q]$ for polynomials P, Q over some ring R denotes the substitution of x with Q in P.

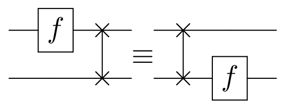
Proposition 2.7. *For any well-formed, compatible path-sums ξ, ξ', $\xi' \circ \xi$ is also well formed. Moreover,*

$$U_{\xi'\circ\xi} = U_{\xi'}U_\xi.$$

[1] Determining compatibility is at least as hard as detecting whether an ancilla is *clean* and is hence non-trivial in general. For the verification tasks we consider this is not an issue, as in practice we only compose path-sums with unitary operators.

Remark 2.8. A useful property of path-sums is that they unify structurally equivalent circuits without resorting to string diagrams, which can be difficult to reason about in automated ways [9]. By this we mean that circuits which are equivalent up to symmetric monoidal laws are *strictly equal* in the path-sum picture. For instance, the bifunctoriality law and the naturality of SWAP, stated respectively as the equivalences





are both equality in the path-sum framework. While much progress has been made towards computational methods for diagrammatic reasoning [9, 11, 13, 15], our framework allows us to use standard algebraic tools (e.g., rewriting) without explicitly managing structural laws.

Along with unifying the representation of *structurally* equivalent circuits, path-sums further unify many *semantic* equivalences of quantum circuits – particularly allowing the long-distance cancellation of phase gates applied to the same logical state [2]. In contrast, matrix representations unify *all* equivalences between unitaries, at the expense of exponential space. Path-sums hence provide an intermediary model, where many equivalences are "modded out" yet still generally remain efficiently representable as we show next.

2.2 Path-sums as a circuit semantics

As path-sums admit both a symmetric tensor product and functional composition, we can give a simple compositional path-sum semantics of measurement-free quantum circuits. Given a path-sum representation of each gate in a basis $\mathcal{B}$ and their inverses, we can define the path-sum interpretation of a circuit over $\mathcal{B}$ as the composition of each gate. In particular, we give a path-sum interpretation to the Clifford+R_k basis $\{H, \mathrm{CNOT}, R_k\}$ for $k > 0$.

Definition 2.9. (Clifford+R_k path-sum)
The path-sum interpretation of an n-qubit circuit C over $\{H, \mathrm{CNOT}, R_k\}$, denoted $[\![C]\!]$, is defined as follows:

$$\begin{aligned}
[\![H]\!] &= |x\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}} e^{2\pi i \frac{xy}{2}} |y\rangle \\
[\![R_k]\!] &= |x\rangle \mapsto e^{2\pi i \frac{x}{2^k}} |x\rangle \\
[\![R_k^\dagger]\!] &= |x\rangle \mapsto e^{2\pi i \frac{-x}{2^k}} |x\rangle \\
[\![\mathrm{CNOT}]\!] &= |x_1 x_2\rangle \mapsto |x_1 (x_1 \oplus x_2)\rangle \\
[\![C_1; C_2]\!] &= [\![C_2]\!] \circ [\![C_1]\!].
\end{aligned}$$

We leave the appropriate vertical compositions implicit.

Proposition 2.10. *For any circuit C over $\{H, \mathrm{CNOT}, R_k\}$ with unitary matrix U_C, we have $U_{[\![C]\!]} = U_C$.*

As a composition of linear Boolean functions, it can trivially be observed that each of the outputs of a canonical path-sum is linear. Moreover, its phase polynomial has degree at most k. To show this, we first introduce the notion of the *order* of a polynomial in $\mathbb{D}_M$ which gives a more precise characterization of the phase polynomials over a fixed level of the Clifford hierarchy. Note that without loss of generality we can restrict our attention to phase polynomials with coefficients in $\mathbb{D}/\mathbb{Z}$ since $e^{2\pi i} = 1$.

Definition 2.11. The *order* of a term $\frac{a}{2^b}\mathbf{x}^\alpha$ where a is co-prime to 2 and $\alpha \in \mathbb{Z}_2^n$ is $b+|\alpha|-1$. The order of a polynomial $P \in \mathbb{D}_M[\mathbf{x}]$, denoted $\mathsf{ord}\,(P)$, is the maximum order of all terms in P.

Example 2.12.

$$\mathsf{ord}\left(\frac{1}{2}\right)=0, \qquad \mathsf{ord}\left(\frac{1}{2}x_1+\frac{1}{2}x_2\right)=1, \qquad \mathsf{ord}\left(\frac{1}{2^3}x_2+\frac{1}{2}x_1x_2x_3\right)=3$$

An important fact, shown below, is that order is non-increasing with respect to substitution of linear Boolean polynomials.

Lemma 2.13. *Let $P \in \mathbb{D}_M[\mathbf{x}]$, and let $Q \in \mathbb{Z}_2[\mathbf{x}]$ be a linear polynomial. Then for any x_i,*

$$\mathsf{ord}\left(P[x_i \leftarrow \overline{Q}]\right) \leq \mathsf{ord}\,(P)$$

Proof. Suppose $Q=\sum_{j\in S} x_j$ for some set S. It is easy to verify that

$$\overline{\sum_{j\in S} x_j} = \sum_{S'\subseteq S} (-2)^{|S'|-1} \prod_{j\in S'} x_j.$$

Substituting $\overline{Q}$ in for x_i we see that for any term $\frac{a}{2^b}\mathbf{x}^\alpha$ in P such that $\alpha_i=1$,

$$\begin{aligned}\mathsf{ord}\left(\frac{a}{2^b}\mathbf{x}^\alpha[x_i \leftarrow \overline{Q}]\right) &= \max_{S'\subseteq S}\mathsf{ord}\left(\frac{a2^{|S'|-1}}{2^b}\mathbf{x}^\alpha[x_i \leftarrow \prod_{j\in S'} x_j]\right) \\ &\leq \max_{S'\subseteq S} b-|S'|+|\alpha|+|S'|-1 \\ &= \mathsf{ord}\left(\frac{a}{2^b}\mathbf{x}^\alpha\right).\end{aligned}$$

□

Intuitively, since the output function of a Clifford+R_k path-sum is strictly linear, composing Clifford+R_k path-sums does not increase the order of the phase polynomial. Moreover, the path-sum interpretation of each gate over $\{H, \mathrm{CNOT}, R_k\}$ has a phase polynomial of order at most k and maximum denominator 2^k, hence we obtain the following result.

Proposition 2.14. *The phase polynomial of a (canonical) Clifford+R_k path-sum has degree at most k.*

Corollary 2.15. *The path-sum interpretation of an n-qubit Clifford+R_k circuit C has size polynomial in the volume of C ($n\cdot|C|$) and can be computed in polynomial time.*

On representations of the phase polynomial While the representation of the phase as a multilinear polynomial is indeed polynomial in the size of the circuit, at higher levels of the Clifford hierarchy (i.e. large k) the degree of the polynomial can become prohibitively large. Even for the standard Clifford+T gate set, the path-sum of a circuit requires space cubic in the volume of the circuit [4]. In practice this makes verification of some larger circuits difficult.

The phase polynomial could instead be represented in *linear space for any k* by its *Fourier expansion* [1,26]. This however complicates the process of verification as the Fourier expansion is not necessarily unique modulo integer multiples [1]. A possible compromise would be to store the Fourier expansion normally, and generate the multilinear form for small subsets on demand.

3 A calculus for path-sums

The verification question we're generally concerned with is *given a circuit C and path-sum ξ, is $[\![C]\!] \equiv \xi$?.* From an automated perspective it is simpler to instead check that the path-sum *miter* [31] $[\![C^\dagger]\!] \circ \xi$ is the identity transformation. In either case, we need a method of efficiently establishing equivalence. To that end, in this section we present a system of reduction rules for path-sums. A key feature of our calculus is that the reduction rules strictly decrease the number of path variables, producing a (not necessarily unique) normal form in polynomial time.

3.1 Overview

Our calculus operates by reducing the number of paths when sets of paths interfere in recognizable ways which we call *interference patterns.* As an illustration, consider the identity circuit HH. Computing its canonical path-sum we get

$$HH : |x\rangle \mapsto \frac{1}{\sqrt{2^2}} \sum_{y_1, y_2 \in \mathbb{Z}_2} e^{2\pi i \frac{xy_1 + y_1 y_2}{2}} |y_2\rangle.$$

To see that the above path-sum is equal to the identity, we can first expand the exponential sum on the right by the values of the internal path variable y_1:

$$\frac{1}{\sqrt{2^2}} \sum_{y_1, y_2 \in \mathbb{Z}_2} e^{2\pi i \frac{xy_1 + y_1 y_2}{2}} |y_2\rangle = \frac{1}{\sqrt{2^2}} \sum_{y_2 \in \mathbb{Z}_2} (1 + e^{2\pi i \frac{x + y_2}{2}}) |y_2\rangle$$

Since $e^\pi i = -1$, it can be observed that if $x + y_2 = 0 \mod 2$, the two paths corresponding to $y_1 = 0$ and $y_2 = 1$ *constructively* interfere, whereas if $x + y_2 = 1 \mod 2$ they *destructively* interfere. As $\mathbb{Z}_2 = x \oplus \mathbb{Z}_2 = \{x, 1 \oplus x\}$ for any $x \in \mathbb{Z}$, we can rewrite the sum over $x \oplus \mathbb{Z}_2$ and explicitly calculate the interference on either path:

$$\begin{aligned} \frac{1}{\sqrt{2^2}} \sum_{y_2 \in x \oplus \mathbb{Z}_2} (1 + e^{2\pi i \frac{x + y_2}{2}}) |y_2\rangle &= \frac{1}{2}(1 + e^{2\pi i \frac{x+x}{2}})|x\rangle + \frac{1}{2}(1 + e^{2\pi i \frac{x+1+x}{2}})|1 \oplus x\rangle \\ &= \frac{2}{2}|x\rangle + \frac{0}{2}|1 \oplus x\rangle \\ &= |x\rangle \end{aligned}$$

The reasoning above applies to any situation where an internal path variable y_i only appears with coefficients taken from the Boolean subgroup $\{0, \frac{1}{2}\}$ of $\mathbb{D}/\mathbb{Z}$, as the two branches of y_i are identical, except that $y_i = 1$ path picks up a multiplicative factor of -1 whenever the quotient of P/y_i is odd. Specifically, it can be shown that

$$\frac{1}{\sqrt{2^{m+1}}} \sum_{y_0 \in \mathbb{Z}_2} \sum_{\mathbf{y} \in \mathbb{Z}_2^m} e^{2\pi i \left(\frac{1}{2} y_0 Q(\mathbf{x}, \mathbf{y}) + R(\mathbf{x}, \mathbf{y})\right)} |f(\mathbf{x}, \mathbf{y})\rangle = \frac{1}{\sqrt{2^{m-1}}} \sum_{\mathbf{y} \in \mathbb{Z}_2^m, Q(\mathbf{x}, \mathbf{y}) = 0 \mod 2} e^{2\pi i R(\mathbf{x}, \mathbf{y})} |f(\mathbf{x}, \mathbf{y})\rangle$$

Note that the polynomial $Q(\mathbf{x}, \mathbf{y})$ is Boolean-valued, as otherwise the $y_0 = 1$ path can pick up values not in $\{1, -1\}$. In practice, we only perform such reductions when the restricted sum can be reified by solving $Q(\mathbf{x}, \mathbf{y}) = 0 \mod 2$ for some path variable, as we did above with $y_2 = x$.

$$\frac{1}{\sqrt{2^{m+2}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle \longrightarrow \frac{1}{\sqrt{2^{m}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle \qquad \textsf{[Elim]}$$

$$\frac{1}{\sqrt{2^{m+1}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{4}y_0+\frac{1}{2}y_0Q(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle \longrightarrow \frac{1}{\sqrt{2^{m}}}\sum_{y\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{8}-\frac{1}{4}\overline{Q}(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle \qquad [\omega]$$

$$\frac{1}{\sqrt{2^{m+1}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{2}y_0(y_i+Q(\mathbf{x},\mathbf{y}))+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle \longrightarrow \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(R[y_i\leftarrow\overline{Q}]\right)(\mathbf{x},\mathbf{y})}|\left(f[y_i\leftarrow Q]\right)(\mathbf{x},\mathbf{y})\rangle \qquad \textsf{[HH]}$$

$$\frac{P(\mathbf{x},\mathbf{y})=\frac{1}{4}y_ix+\frac{1}{2}y_i(y_j+Q(\mathbf{x},\mathbf{y}))+R(\mathbf{x},\mathbf{y})=\frac{1}{4}y_j(1-x)+\frac{1}{2}y_j(y_i+Q'(\mathbf{x},\mathbf{y}))+R'(\mathbf{x},\mathbf{y})}{\frac{1}{\sqrt{2^{m+2}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m+2}} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle \longrightarrow \frac{1}{\sqrt{2^m}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left((1-x)R[y_j\leftarrow\overline{Q}]+xR'[y_i\leftarrow\overline{Q'}]\right)(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle} \qquad \textsf{[Case]}$$

Figure 2: Path-sum reduction rules

3.2 Reduction rules

Figure 2 gives the rules of our calculus, presented as algebraic rewrite rules on exponential sums for convenience and applied to path-sums in the obvious way. We write $\xi \longrightarrow \xi'$ to denote that ξ reduces to ξ', and denote by $\longrightarrow^*$ the transitive closure of $\longrightarrow$. For all rules, y_0 is an internal path variable, quotients Q are Boolean-valued and whenever $y_i \leftarrow Q$, y_i does not appear in Q. For the [Case] rule, both y_i and y_j are internal.

The rules were developed by translating known circuit identities into path-sums, then minimizing the identities to obtain simple interference patterns which 1) strictly reduce the number of path variables, and 2) can be efficiently matched. What we found was that most common Clifford+T equalities reduce to a small set of rules – in particular, the [HH] rule derived from the equality $HH = I$ as described above is sufficient for the vast majority of path-sum reductions. The $[\omega]$ rule arises from the identity $(SH)^3 = e^{\frac{2\pi i}{8}}I$, and the final rule [Case] is a specific case distinction needed to prove the 2-qubit Clifford+T identity $\left(\mathrm{CNOT}(X\otimes T)\textsf{controlled-}H(X\otimes T^\dagger)\right)^2$ [29]. The [Elim] rule only appears to simplify the presentation of [HH] as well as in some contexts specific to verification which we describe later.

Proposition 3.1 (Correctness)**.** *If $\xi \longrightarrow^* \xi''$, then $\xi \equiv \xi'$.*

The correctness of our rewrite system follows from direct calculation over symbolic exponential sums. As the proof is quite tedious, we leave it to Appendix A.

It is a trivial fact that our calculus is terminating, as every rule reduces the number of path variables. Moreover, each rewrite rule can be matched against in polynomial time, hence every path-sum reduces to a normal form in polynomial time.

Proposition 3.2 (Strong normalization)**.** *Every sequence of rewrites terminates with an irreducible path-sum. The sequence is linear in the number of path variables m and for an n-qubit path-sum takes time polynomial in n and m.*

3.3 Examples

To illustrate our rewrite system, we give examples below. Further examples can be found in Appendix B.

Example 3.3. Recall that the standard implementation of the Toffoli gate over Clifford+T has the path-sum form [2]

$$\mathsf{Toffoli}_3 : |x_1x_2x_3\rangle \mapsto \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2\in\mathbb{Z}_2} e^{2\pi i\frac{1}{2}(x_3y_1+x_1x_2y_1+y_1y_2)}|x_1x_2y_2\rangle.$$

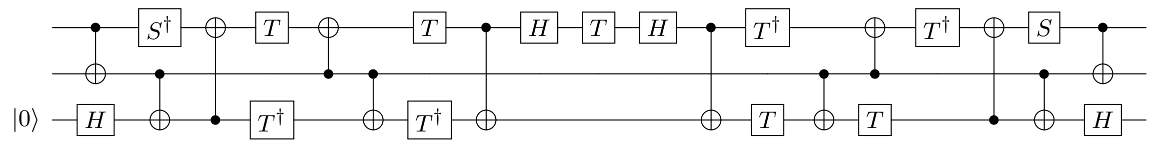
We can verify that this is equivalent to the functional specification $|x_1x_2x_3\rangle \mapsto |x_1x_2(x_3\oplus x_1x_2)\rangle$ with the following sequence of reductions and algebraic manipulations:

$$\begin{aligned}|x_1x_2x_3\rangle &\mapsto \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2\in\mathbb{Z}_2} e^{2\pi i\frac{1}{2}(x_3y_1+x_1x_2y_1+y_1y_2)}|x_1x_2y_2\rangle \\ &\mapsto \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2\in\mathbb{Z}_2} e^{2\pi i\frac{1}{2}y_1(y_2+x_3+x_1x_2)}|x_1x_2y_2\rangle \\ &\mapsto \frac{1}{\sqrt{2^2}} \sum_{y_2\in\mathbb{Z}_2} |x_1x_2(x_3\oplus x_1x_2)\rangle && \text{[HH]} \\ &\mapsto |x_1x_2(x_3\oplus x_1x_2)\rangle && \text{[Elim]}\end{aligned}$$

Example 3.4. The controlled-T gate can be specified as the path-sum

$$\mathsf{controlled\text{-}T} : |x_1x_2\rangle \mapsto e^{2\pi i\frac{x_1x_2}{8}}|x_1x_2\rangle.$$

An implementation of the controlled-T gate over Clifford+T is given below:



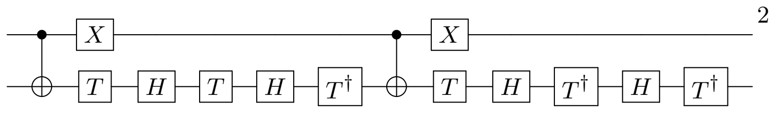
Computing the canonical path-sum and reducing we get

$$\begin{aligned}|x_1x_2\rangle|0\rangle &\mapsto \frac{1}{\sqrt{2^4}} \sum_{\mathbf{y}\in\mathbb{Z}_2^4} e^{2\pi i\frac{1}{8}(4x_1x_2y_1+4x_1y_2+4y_1y_2+y_2+4y_2y_3+4x_1x_2y_3+4x_1y_4+4y_3y_4+4x_1x_2)}|x_1x_2y_4\rangle \\ &\mapsto \frac{1}{\sqrt{2^4}} \sum_{\mathbf{y}\in\mathbb{Z}_2^4} e^{2\pi i\left(\frac{1}{2}y_1(y_2+x_1x_2)+\frac{1}{8}(4x_1y_2+y_2+4y_2y_3+4x_1x_2y_3+4x_1y_4+4y_3y_4+4x_1x_2)\right)}|x_1x_2y_4\rangle \\ &\mapsto \frac{1}{\sqrt{2^2}} \sum_{y_3,y_4\in\mathbb{Z}_2} e^{2\pi i\frac{1}{8}(4x_1x_2+x_1x_2+4x_1x_2y_3+4x_1x_2y_3+4x_1y_4+4y_3y_4+4x_1x_2)}|x_1x_2y_4\rangle && \text{[HH, Elim]} \\ &\mapsto \frac{1}{\sqrt{2^2}} \sum_{y_3,y_4\in\mathbb{Z}_2} e^{2\pi i\left(\frac{1}{2}y_3y_4+\frac{1}{8}(x_1y_4+x_1x_2)\right)}|x_1x_2y_4\rangle \\ &\mapsto e^{2\pi i\frac{x_1x_2}{8}}|x_1x_2\rangle|0\rangle && \text{[HH, Elim]}\end{aligned}$$

Hence the above circuit implements the controlled-T gate, and provably leaves the ancilla clean.

4 Completeness

While our calculus computes a normal form in polynomial time, the normal forms are not necessarily unique[2] and hence our reduction system is incomplete. For instance, the Clifford+T identity



from [29] gives the irreducible path-sum $|x_1x_2\rangle \mapsto \frac{1}{\sqrt{2^8}}\sum_{\mathbf{y}\in\mathbb{Z}_2^8} e^{2\pi i\frac{1}{8}P(\mathbf{x},\mathbf{y})}|x_1y_8\rangle$ with phase polynomial

$$\begin{aligned}P(\mathbf{x},\mathbf{y}) = 2 &+ 6x_1x_2 + x_2 + y_1 + 4y_1(x_1+x_2+y_2) + 6y_2 + 4y_2y_3 + 2y_2x_1 + 3y_3 + 4y_3(x_1+y_4)\\ &+ 4y_4y_5 + 6y_4x_1 + y_5 + 4y_5(x_1+y_6) + 6y_6 + 4y_6y_7 + 2y_6x_1 + 3y_7 + 4y_7(x_1+y_8) + 7y_8.\end{aligned}$$

A complete verification procedure could proceed by explicitly expanding the values of remaining variables in the path-sum after all possible reductions have been made, and then checking equivalence to the identity transformation. In practice we found that this is generally not necessary, as our calculus, along with some additional observations, is sufficient to prove equivalence or non-equivalence for the majority of circuits. Moreover, these heuristics combined with path-sum reductions give a complete, polynomial-time procedure for determining equivalence of Clifford group circuits.

4.1 Isometry restrictions

Our first heuristic reduces the number of path variables in a *well-formed* path sum when checking equivalence. Specifically, we denote by $\xi|_{f(\mathbf{x},\mathbf{y})=\mathbf{x}}$ the restriction of ξ to solutions $\mathbf{x}\in\mathbb{Z}_2^n, \mathbf{y}\in\mathbb{Z}_2^m$ such that $f(\mathbf{x},\mathbf{y})=\mathbf{x}$, which we can write as the restricted sum

$$|\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^m}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m, f(\mathbf{x},\mathbf{y})=\mathbf{x}} e^{2\pi i P(\mathbf{x},\mathbf{y})}|\mathbf{x}\rangle.$$

Effectively, the sum $\frac{1}{\sqrt{2^m}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m, f(\mathbf{x},\mathbf{y})=\mathbf{x}} e^{2\pi i P(\mathbf{x},\mathbf{y})}$ gives the amplitude of the basis state $|\mathbf{x}\rangle$ in the output for a given input state $|\mathbf{x}\rangle$. If the path sum ξ is well-formed (i.e. isometric), then this sum will be equal to 1 exactly if ξ is the identity transformation. We sum this up in the lemma below:

Lemma 4.1. *Suppose $U_\xi : |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^m}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$ is a well-formed path-sum. Then $\xi \equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$ if and only if $\xi|_{f(\mathbf{x},\mathbf{y})=\mathbf{x}} \equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$.*

Note that lemma 4.1 doesn't hold if ξ is not well-formed, as U_ξ may not be an isometry and so it may be that $U_\xi|\mathbf{x}\rangle = |\mathbf{x}\rangle + |\psi\rangle$ for some residual state $|\psi\rangle$. To reify the restricted path-sum $\xi|_{f(\mathbf{x},\mathbf{y})=\mathbf{x}}$ we find path variable substitutions which give $f_i(\mathbf{x},\mathbf{y}) = x_i$ – in particular, if for some index i we have $f_i(\mathbf{x},\mathbf{y}) = y_i \oplus Q(\mathbf{x},\mathbf{y})$ where y_i doesn't appear in $Q(\mathbf{x},\mathbf{y})$, we can substitute $Q(\mathbf{x},\mathbf{y})$ for y_i to get $f_i(\mathbf{x},\mathbf{y}) = x_i$ and remove y_i from the sum. Any restrictions which can't be reified are simply ignored. In practice this results in a significant simplification for some circuits, instantly removing up to n path variables.

[2]It was pointed out by an anonymous referee that uniqueness would imply that equivalence checking of reversible Boolean circuits is in P. As this problem is co-NP-complete, uniqueness of our normal forms would indeed imply P = co-NP.

4.2 Non-equivalence

As the reduction rules of fig. 2 only suffice to prove *positive* results, when no more reductions are possible we apply an observation that was found to be effective for proving that a path sum ξ is *not* the identity. In particular, recall that

$$\frac{1}{\sqrt{2^{m+1}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{2}y_0 Q(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle = 0$$

if $Q(\mathbf{x},\mathbf{y}) = 1 \mod 2$. If Q is a non-zero Boolean-valued polynomial in *only* input variables x_i, then there necessarily exists a solution $\mathbf{x}\in\mathbb{Z}^n$ such that $Q(\mathbf{x}) = 1 \mod 2$ [26], and in particular

$$\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{2}y_0 Q(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle = \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m}(1-1)e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle = 0.$$

We sum this up in the following lemma.

Lemma 4.2. *Suppose* $U_\xi : |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^{m+1}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{2}y_0 Q(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle$ *where* Q *is a non-zero integer-valued polynomial not containing any path variables. Then* $\xi \not\equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$.

Hence we can use a variant of [HH] where Q contains only input variables to prove non-equivalence of a path-sum to the identity.

4.3 Clifford completeness

We can now show that together with the above simplifications, our path-sum reductions are complete for proving equivalence of Clifford group circuits. Recall that over the Clifford group, the path-sum interpretation of a circuit has phase polynomial of order at most 2. Our proof of completeness rests on the fact that progress can always be made for an identity path-sum with only internal path variables and second-order phase polynomial, as shown below.

Lemma 4.3 (Clifford progress & preservation)**.** *If* ξ *is a path-sum such that* $\xi \equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$, $\mathsf{ord}\,(P) \leq 2$ *and* ξ *contains only internal path variables, then there exists* ξ' *such that* $\xi \longrightarrow \xi'$ *and* $\mathsf{ord}\,(P') \leq 2$.

Proof. Since P is at most second-order, we can write $P = y_0 Q + R$ for some internal path variable y_0 and polynomials Q, R where Q is at most first-order, and in particular has the form

$$a\frac{1}{4} + b\frac{1}{2}Q'$$

where $a, b \in \mathbb{Z}_2$ and Q' is a linear Boolean-valued polynomial. We have 3 cases to consider, corresponding to the [Elim], [HH] and [ω] rules respectively.

Case 1: $a = b = 0$**.** The variable y_i does not appear in P, hence $\xi \longrightarrow_{[\mathsf{Elim}]} \xi'$ and $\mathsf{ord}\,(P') = \mathsf{ord}\,(P) \leq 2$.

Case 2: $a = 0, b = 1$**.** If the polynomial Q' contains a path variable y_i, then $Q' = y_i + Q''$ and $\xi \longrightarrow_{[\mathsf{HH}]} \xi'$. Further, by lemma 2.13, $\mathsf{ord}\left(R[y_i \leftarrow \overline{Q''}]\right) \leq \mathsf{ord}\,(R) \leq 2$ and ξ' has only internal paths since $y_i \notin f$.

If on the other hand Q' only contains input variables, by lemma 4.2 $\xi \not\equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$, a contradiction.

Case 3: $a = 1$. The sum matches the left hand side of $[\omega]$, hence $\xi \longrightarrow_{[\omega]} \xi'$. Further, by lemma 2.13

$$\mathsf{ord}\,(P') = \mathsf{ord}\left(\frac{1}{8} - \frac{1}{4}\overline{Q''} + R\right) = \max\left\{\mathsf{ord}\left(\frac{1}{8}\right), \mathsf{ord}\left(\frac{1}{4}\overline{Q''}\right), \mathsf{ord}\,(R)\right\} = 2.$$

$\square$

Corollary 4.4. *If C is a Clifford-group quantum circuit, then $[\![C]\!] \equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$ can be decided in time polynomial in the space-time volume of C.*

Proof. Since $[\![C]\!]$ is well-formed, by lemma 4.1 it suffices to check $[\![C]\!]|_{f(\mathbf{x},\mathbf{y})=\mathbf{x}} \equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$. Further, as $f(\mathbf{x},\mathbf{y})$ is linear, we can compute via Gaussian elimination a solution $\mathbf{y}$ so that $f(\mathbf{x},\mathbf{y}) = \mathbf{x}$ for any $\mathbf{x}$ – if no such solution exists, $[\![C]\!] \not\equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$. Since each f_i is linear, $\mathsf{ord}\,(P[y_i \leftarrow f_i]) \leq \mathsf{ord}\,(P) \leq 2$, hence by lemma 4.3 and proposition 3.2, either $[\![C]\!]|_{f(\mathbf{x},\mathbf{y})=\mathbf{x}}$ reduces to $|\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$ in polynomial-time or $\xi' \not\equiv |\mathbf{x}\rangle \mapsto |\mathbf{x}\rangle$. $\square$

5 Case studies

We implemented our framework and verification algorithm in the open-source Haskell library Feynman. To test the efficacy of our methods, we performed verification of circuit optimizations (both correct and incorrect), as well as the verification of circuit implementations against formal path-sum specifications. All experiments were run in Debian Linux running on a quad-core 64-bit Intel Core i7 2.40 GHz processor and 8 GB RAM, and can be executed from the command line with `./feyn VerBench` and `./feyn VerAlg` for the translation validation and algorithm benchmarks, respectively.

5.1 Translation validation

Translation validation is an important tool for verifying that the transformations a compiler performs do not change the semantics of an input program. While it is generally desirable to prove that a compiler operates correctly on *all* input programs, as with *verified compilers* like CompCert [22] or ReVerC [5] in the reversible domain, in many cases this is infeasible since the best optimizations are typically difficult to formally verify.

We used our algorithm to verify a suite of optimized benchmark circuits against their original input. For the optimization algorithm we chose the GraySynth algorithm from [1] which is implemented in Feynman and verified each benchmark reported in that paper. Table 1 reports the results of our experiments. All but 3 of the benchmark circuits were successfully verified, with the remaining 3 benchmarks running out of memory with a 6 GB limit. The high memory usage may be mitigated in the future by switching to a linear-space representation of the phase polynomial. The largest (completed) benchmark GF(2^{32}), containing 96 bits, 252 path variables and over 25000 gates completed in under 10 minutes, with the remainder all taking under a minute.

To test the algorithm's ability to prove *non-equivalence*, we also performed the verification of the optimized benchmark circuits after removing a randomly selected gate. Again, all but 3 benchmarks were proven to be not equivalent, with the negative verification results taking about the same amount of time as positive results.

Table 1: Translation validation results. n lists the number of qubits, Path vars gives the number of path variables, and Clifford and T give the number of respective gates. Times for positive and negative verification measure the time to prove equivalence or non-equivalence against the optimized circuit or the optimized circuit with one random gate removed, respectively. Benchmarks with no timing results ran out of memory.

Algorithm	n	Path vars	Clifford	T	Time (s)	
					Positive	Negative
Grover_5	9	200	1515	490	0.973	0.988
Mod 5_4	5	12	66	44	0.005	0.028
VBE-Adder_3	10	20	167	94	0.026	0.028
CSLA-MUX_3	15	40	289	132	0.099	0.055
CSUM-MUX_9	30	56	638	280	0.270	0.270
QCLA-Com_7	24	74	1237	297	0.530	0.543
QCLA-Mod_7	26	164	1641	650	9.446	10.517
QCLA-Adder_10	36	100	627	400	0.674	0.683
Adder_8	24	160	1419	614	1.968	2.018
RC-Adder_6	14	44	322	124	0.080	0.090
Mod-Red_21	11	60	392	192	0.110	0.119
Mod-Mult_55	9	28	180	84	0.028	0.009
Mod-Adder_1024	28	660	4363	3006	21.362	21.588
Cycle 17_3	35	1366	9172	6694	–	–
GF(2^4)-Mult	12	28	263	180	0.063	0.061
GF(2^5)-Mult	15	36	393	286	0.143	0.141
GF(2^6)-Mult	18	44	559	402	0.279	0.291
GF(2^7)-Mult	21	52	731	560	0.501	0.527
GF(2^8)-Mult	24	60	975	712	0.837	0.881
GF(2^9)-Mult	27	68	1179	918	1.304	1.369
GF(2^{10})-Mult	30	76	1475	1110	1.958	0.327
GF(2^{16})-Mult	48	124	3694	2832	16.028	17.539
GF(2^{32})-Mult	96	252	14259	11296	430.883	436.521
GF(2^{64})-Mult	192	508	55408	45120	–	–
Hamming_15 (low)	17	76	612	158	0.367	0.168
Hamming_15 (med)	17	184	1251	762	1.390	1.430
Hamming_15 (high)	20	716	5332	3462	24.360	24.303
HWB_6	7	52	369	180	0.200	0.207
HWB_8	12	2282	17583	8895	–	–
QFT_4	5	84	218	136	0.084	0.089
$\Lambda_3(X)$	5	12	52	36	0.004	0.011
$\Lambda_3(X)$ (Barenco)	5	12	66	44	0.007	0.046
$\Lambda_4(X)$	7	20	87	58	0.009	0.008
$\Lambda_4(X)$ (Barenco)	7	20	127	84	0.014	0.024
$\Lambda_5(X)$	9	18	112	80	0.015	0.017
$\Lambda_5(X)$ (Barenco)	9	28	160	124	0.030	0.031
$\Lambda_{10}(X)$	19	68	297	190	0.110	0.111
$\Lambda_{10}(X)$ (Barenco)	19	68	493	324	0.219	0.210

Table 2: Results of verifying formally specified quantum algorithms.

Algorithm	n	Path vars	Clifford	T	Time (s)	
					Positive	Negative
$\mathsf{Toffoli}_{50}$	97	190	855	665	1.084	1.064
$\mathsf{Toffoli}_{100}$	197	390	1755	1365	5.566	5.275
Maslov_{50}	74	192	481	384	0.801	0.778
Maslov_{100}	149	392	981	784	3.987	3.983
Adder_{8}	40	56	334	196	0.142	0.143
Adder_{16}	80	120	710	420	25.527	92.607
QFT_{16}	16	16	256	–	1.250	1.335
QFT_{31}	31	31	961	–	16.929	15.295
$\mathsf{Hidden\ Shift}_{20,4}$	20	60	5254	56	1.067	0.862
$\mathsf{Hidden\ Shift}_{40,5}$	40	120	6466	70	3.383	2.826
$\mathsf{Hidden\ Shift}_{60,10}$	60	180	12784	140	13.217	12.351
$\mathsf{Symbolic\ Shift}_{20,4}$	40	60	5296	56	1.859	1.849
$\mathsf{Symbolic\ Shift}_{40,5}$	80	120	6638	70	6.953	7.905
$\mathsf{Symbolic\ Shift}_{60,10}$	120	180	12804	140	35.583	29.614

5.2 Verifying quantum algorithms

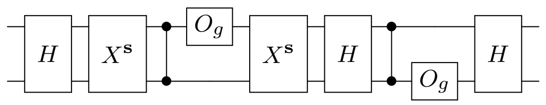
To evaluate our framework as a tool for functional specification and verification, we implemented and verified several quantum algorithms (both without and with errors) directly against their specification as a path sum. Table 2 reports the results of our experiments, and we describe the algorithms and implementations below.

Reversible functions We implemented and verified a number of known algorithms for reversible functions. In particular, we performed verifications of Clifford+T implementations of the generalized Toffoli and (out-of-place) addition functions,

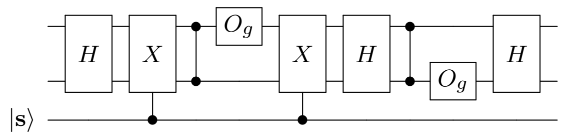
$$\begin{aligned}\mathsf{Toffoli}_n &: |x_1x_2\dots x_n\rangle \mapsto |x_1x_2\dots(x_n \oplus x_1x_2\dots x_{n-1})\rangle, \\ \mathsf{Adder}_n &: |\mathbf{x}\rangle|\mathbf{y}\rangle|\mathbf{0}\rangle \mapsto |\mathbf{x}\rangle|\mathbf{y}\rangle|\mathbf{x}+\mathbf{y}\rangle\end{aligned}$$

We chose two implementations of the n-bit Toffoli gate – using the standard decomposition into $2(n-3)+1$ Toffoli gates and $n-3$ ancillas, and the Maslov decomposition [23] using *relative phase* Toffolis and $\lceil\frac{n-3}{2}\rceil$ ancillas. For either implementation we were able to verify up to 100 bit Toffoli gates in just seconds.

For the addition circuit, we used a standard out-of-place ripple-carry adder which uses $n-1$ ancilla bits to store intermediate carry values and an additional n bit register to store the output, before copying out and uncomputing. The resulting circuit uses $5n-1$ bits of space for an n bit adder, and $4(n-1)$ Toffoli gates, which are then expanded to the Clifford+T gate set. The specification itself was generated by implementing binary addition on symbolic vectors, and could ostensibly be classically tested to verify its own correctness. In this case, the size of the bitwise expansion of $\mathbf{x}+\mathbf{y}$ made it difficult to push to implementation sizes (e.g., 32 bits), though smaller sizes such as 16 bits were verifiable within a minute. *Relational* techniques – e.g., representing the outputs of a path-sum as "primed" variables along with equations relating them – may help to push verification of such functions to larger sizes.



(a) Hidden shift with a fixed shift $\mathbf{s}$.



(b) Hidden shift with a symbolic shift.

Figure 3: Circuits for the Quantum Hidden Shift algorithm.

The quantum Fourier transform To test our verification method against circuits using higher-order rotations, we verified an implementation of the quantum Fourier transform. We use a circuit from [20] together with a final qubit permutation correction and verified it against the specification

$$\mathsf{QFT}_n : |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{\mathbf{y}\in\mathbb{Z}_2^n} e^{2\pi i \frac{[\mathbf{x}\cdot\mathbf{y}]}{2^n}} |\mathbf{y}\rangle.$$

The phase polynomial $[\mathbf{x}\cdot\mathbf{y}]$ was generated in the obvious way – by computing $[\mathbf{x}] = x_1 + 2x_2 + \ldots + 2^{n-1}x_n$ and multiplying the polynomials. In this case our implementation was able to verify implementations up to 31 bits in size, after which integer overflow occurs due to our handling of dyadic arithmetic. Given that the 31 bit implementation took only 16 seconds to verify, it appears that with better methods for handling dyadic arithmetic much larger sizes of the QFT are likely verifiable.

The quantum hidden shift algorithm To test our framework on more general quantum algorithms, we implemented a version of the quantum hidden shift algorithm [28] which has been previously used to test quantum simulation algorithms [10]. In particular, given oracles $O_{f'} : |\mathbf{x}\rangle \mapsto f(\mathbf{x}+\mathbf{s})|\mathbf{x}\rangle$ and $O_{\tilde{f}} : |\mathbf{x}\rangle \mapsto \tilde{f}(\mathbf{x})|\mathbf{x}\rangle$ for the shifted and dual bent functions $f', \tilde{f} : \mathbb{Z}_2^n \to \{-1,+1\}$ respectively, the circuit $H^{\otimes n} O_{\tilde{f}} H^{\otimes n} O_{f'} H^{\otimes n}$ is known [28] to implement the mapping $|\mathbf{0}\rangle \mapsto |\mathbf{s}\rangle$.

Following [10], we generated random instances of Maiorana McFarland bent functions by setting $f'(\mathbf{x},\mathbf{y}) = f((\mathbf{x},\mathbf{y})+\mathbf{s}) = (-1)^{g(\mathbf{x})+\mathbf{xy}}$ with dual $\tilde{f}(\mathbf{x},\mathbf{y}) = (-1)^{g(\mathbf{y})+\mathbf{xy}}$ for a random $\frac{n}{2}$ bit Boolean function g of degree 3. The circuit for f is generated by, for a given number of alternations A, alternating between selecting 200 random Z and controlled-Z gates, then a random doubly controlled-Z gate, expanded out to Clifford+T. We implemented two versions of the algorithm, one where a concrete shift is given by a randomly generated Boolean vector, and another where the shift is supplied symbolically via a quantum register. In the former case we verify the circuit for a given shift $\mathbf{s}$ against the specification $|\mathbf{0}\rangle \mapsto |\mathbf{s}\rangle$, and in the latter case we verify the specification $|\mathbf{0}\rangle|\mathbf{s}\rangle \mapsto |\mathbf{s}\rangle|\mathbf{s}\rangle$. Figure 3 shows both circuits.

Our verification algorithm actually found a bug in our first implementation, which was a direct implementation of the circuit given in [10]. After reimplementing the circuit based on [28], we were able to verify both versions of the hidden shift algorithm for sizes exceeding those simulated in [10] with only a fraction of the time (seconds versus hours [10]). Our calculus further finds the correct output $|\mathbf{s}\rangle$ or $|\mathbf{s}\rangle|\mathbf{s}\rangle$ even without providing the specification, effectively simulating the algorithm rather than verifying it. Moreover, our implementation is deterministic compared to theirs which is probabilistic and only samples the output distribution, rather than compute it outright. It is interesting to note that their algorithm also uses a similar technique of effectively evaluating the circuit's phase polynomial – however, by including the T gate phases directly in the polynomial and solving *around* them, rather than pushing them into state preparations, we save a massive amount of time for this algorithm. An interesting question for future research is to determine whether there are quantum algorithms which can be simulated more efficiently by their methods.

6 Conclusion

We have described a framework for the representation of partial isometries as sums over a discrete set of paths. As an alternative to matrices, our path-sums admit a symbolic representation using polynomials, for which there exists fixed-parameter polynomial size representations of Clifford+R_k circuits. This allows the efficient computation and representation of the action of such a quantum circuit on an arbitrary basis state. Further, we have given a system of rewrite rules which can be used to reduce path-sums and perform functional verification. Our experiments have shown this to be a powerful framework for verifying large quantum circuits, particularly against formal mathematical specifications of quantum algorithms.

The work we have described here is only a preliminary step towards a fully-automated system of formal specification and verification for quantum circuits, and as such there are many issues for future work to address. One particularly appealing direction is to expand the path-sum framework to more general quantum programs, and to give a concrete syntax so that modular libraries of verified programs may be developed and used. Improvements can be made on the algorithmic side, from using Fourier expansions and relational methods to more efficiently store path-sums, to the use of algebraic decision diagrams or other mathematical tools to complete verification once no more reductions can be made. Another interesting direction, motivated by our experience writing path-sum proofs "by hand," is to implement our framework in an interactive proof assistant, allowing inductive and higher-order proofs over entire families of quantum circuits.

7 Acknowledgements

The author wishes to thank Neil J. Ross and Michele Mosca for stimulating discussions on the topic of path sums, as well as the anonymous referees for their helpful comments on an earlier version. This work was supported in part by Canada's NSERC and CIFAR.

References

[1] Matthew Amy, Parsiad Azimzadeh & Michele Mosca (2018): *On the CNOT-complexity of CNOT-PHASE circuits*. *Quantum Science and Technology*, doi:10.1088/2058-9565/aad8ca. Available at `https://arxiv.org/abs/1712.01859`.

[2] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-Time T-depth optimization of Clifford+T circuits via matroid partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489, doi:10.1109/TCAD.2014.2341953. Available at `https://arxiv.org/abs/1303.2042`.

[3] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent & John Schanck (2016): *Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3.* In: *Proceedings of the 24th Conference on Selected Areas in Cryptography (SAC'16)*, pp. 317–337, doi:10.1007/978-3-319-69453-5_18. Available at `https://arxiv.org/abs/1603.09383`.

[4] Matthew Amy & Michele Mosca (2016): *T-count optimization and Reed-Muller codes.* Available at `https://arxiv.org/abs/1601.07363`.

[5] Matthew Amy, Martin Roetteler & Krysta M. Svore (2017): *Verified Compilation of Space-Efficient Reversible Circuits.* In: *Proceedings of the 29th International Conference on Computer Aided Verification (CAV'17)*, pp. 3–21, doi:10.1007/978-3-319-63390-9_1. Available at `https://arxiv.org/abs/1603.01635`.

[6] Linda Anticoli, Carla Piazza, Leonardo Taglialegne & Paolo Zuliani (2016): *Towards Quantum Programs Verification: From Quipper Circuits to QPMC.* In: *Proceedings of the 8th international Conference on Reversible Computation (RC'16)*, pp. 213–219, doi:10.1007/978-3-319-40578-0_16. Available at `https://arxiv.org/abs/1708.06312`.

[7] Ebrahim Ardeshir-Larijani, Simon J. Gay & Rajagopal Nagarajan (2014): *Verification of Concurrent Quantum Protocols by Equivalence Checking.* In: *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, pp. 500–514, doi:10.1007/978-3-642-54862-8_42. Available at `https://arxiv.org/abs/1312.5951`.

[8] Dave Bacon, Wim van Dam & Alexander Russell (2008): *Analyzing algebraic quantum circuits using exponential sums.* Available at `https://www.cs.ucsb.edu/~vandam/LeastAction.pdf`.

[9] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, PawełSobociński & Fabio Zanasi (2016): *Rewriting Modulo Symmetric Monoidal Structure.* In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pp. 710–719, doi:10.1145/2933575.2935316. Available at `https://arxiv.org/abs/1602.06771`.

[10] Sergey Bravyi & David Gosset (2016): *Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates. Physical Review Letters* 116, p. 250501, doi:10.1103/PhysRevLett.116.250501. Available at `https://arxiv.org/abs/1601.07601`.

[11] Bob Coecke, Ross Duncan, Aleks Kissinger & Quanlong Wang (2016): *Generalised Compositional Theories and Diagrammatic Reasoning*, pp. 309–366. Springer Netherlands, Dordrecht, doi:10.1007/978-94-017-7303-4_10. Available at `https://arxiv.org/abs/1506.03632`.

[12] Christopher M. Dawson, Andrew P. Hines, Duncan Mortimer, Henry L. Haselgrove, Michael A. Nielsen & Tobias J. Osborne (2005): *Quantum computing and polynomial equations over the finite field $\mathbb{Z}_2$. Quantum Information and Computation* 5(2), pp. 102–112, doi:10.26421/QIC5.2. Available at `https://arxiv.org/abs/quant-ph/0408129`.

[13] Ross Duncan & Maxime Lucas (2013): *Verifying the Steane code with Quantomatic.* In: *Proceedings of the 10th International Conference on Quantum Physics and Logic (QPL'13)*, 171, pp. 33–49, doi:10.4204/EPTCS.171.4. Available at `https://arxiv.org/abs/1306.4532`.

[14] Richard P. Feynman & Albert R. Hibbs (1965): *Quantum mechanics and path integrals.* McGraw-Hill.

[15] Liam Garvie & Ross Duncan (2017): *Verifying the Smallest Interesting Colour Code with Quantomatic.* In: *Proceedings of the 14th International Conference on Quantum Physics and Logic (QPL'17)*, 266, pp. 147–163, doi:10.4204/EPTCS.266.10. Available at `https://arxiv.org/abs/1706.02717`.

[16] Simon J. Gay, Rajagopal Nagarajan & Nikolaos Papanikolaou (2008): *QMC: A Model Checker for Quantum Systems.* In: *Proceedings of the 20th International Conference on Computer Aided Verification (CAV'08)*, pp. 543–547, doi:10.1007/978-3-540-70545-1_51. Available at `https://arxiv.org/abs/0704.3705`.

[17] Daniel Gottesman & Isaac L. Chuang (1999): *Quantum Teleportation is a Universal Computational Primitive. Nature* 402(6760), p. 390–393, doi:10.1038/46503. Available at `https://arxiv.org/abs/quant-ph/9908010`.

[18] Markus Grassl, Brandon Langenberg, Martin Roetteler & Rainer Steinwandt (2016): *Applying Grover's Algorithm to AES: Quantum Resource Estimates.* In: *Proceedings of the 7th International Workshop on Post-Quantum Cryptography (PQCrypto'16)*, pp. 29–43, doi:10.1007/978-3-319-29360-8_3. Available at `https://arxiv.org/abs/1512.04965`.

[19] IARPA (2013): *Quantum Computer Science.* Available at `http://www.iarpa.gov/Programs/sso/QCS/qcs.html`.

[20] Phillip Kaye, Raymond Laflamme & Michele Mosca (2007): *An Introduction to Quantum Computing.* Oxford University Press.

[21] Dax Enshan Koh, Mark D Penney & Robert W Spekkens (2017): *Computing quopit Clifford circuit amplitudes by the sum-over-paths technique.* Quantum Information and Computation 17(13&14), pp. 1081–1095, doi:10.26421/QIC17.13-14. Available at `https://arxiv.org/abs/1702.03316`.

[22] Xavier Leroy (2006): *Formal Certification of a Compiler Back-end or: Programming a Compiler with a Proof Assistant.* In: *Proceedings of the 34th International Symposium on Principles of Programming Languages (POPL'06)*, ACM, pp. 42–54, doi:10.1145/1111037.1111042.

[23] Dmitri Maslov (2016): *Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization.* Physical Review A 93, p. 022311, doi:10.1103/PhysRevA.93.022311. Available at `https://arxiv.org/abs/1508.03273`.

[24] Ashley Montanaro (2017): *Quantum circuits and low-degree polynomials over $\mathbb{F}_2$.* Journal of Physics A: Mathematical and Theoretical 50(8), p. 084002, doi:10.1088/1751-8121/aa565f. Available at `https://arxiv.org/abs/1607.08473`.

[25] Michael A. Nielsen & Isaac L. Chuang (2000): *Quantum Computation and Quantum Information.* Cambridge University Press.

[26] Ryan O'Donnell (2014): *Analysis of Boolean Functions.* Cambridge University Press, doi:10.1017/CBO9781139814782.

[27] Robert Rand, Jennifer Paykin & Steve Zdancewic (2017): *QWIRE Practice: Formal Verification of Quantum Circuits in Coq.* In: *Proceedings of the 14th International Conference on Quantum Physics and Logic (QPL'17)*, 266, pp. 119–132, doi:10.4204/EPTCS.266.8. Available at `https://arxiv.org/abs/1803.00699`.

[28] Martin Rötteler (2010): *Quantum Algorithms for Highly Non-linear Boolean Functions.* In: *Proceedings of the 21st International Symposium on Discrete Algorithms (SODA'10)*, pp. 448–457, doi:10.1137/1.9781611973075.37. Available at `https://arxiv.org/abs/0811.3208`.

[29] Peter Selinger & Xiaoning Bian (2016): *Relations for 2-qubit Clifford+T operator group.* Available at `https://www.mathstat.dal.ca/~xbian/talks/slide_cliffordt2.pdf`.

[30] Robert Wille, Daniel Grosse, D. Michael Miller & Rolf Drechsler (2009): *Equivalence Checking of Reversible Circuits.* In: *Proceedings of the 39th International Symposium on Multiple-Valued Logic (ISMVL'09)*, pp. 324–330, doi:10.1109/ISMVL.2009.19.

[31] Shigeru Yamashita & Igor L. Markov (2010): *Fast Equivalence-checking for Quantum Circuits.* Quantum Information and Computation 10(9), pp. 721–734, doi:10.26421/QIC10.9-10. Available at `https://arxiv.org/abs/0909.4119`.

[32] Mingsheng Ying (2012): *Floyd–Hoare Logic for Quantum Programs.* ACM Transactions on Programming Languages and Systems 33(6), pp. 19:1–19:49, doi:10.1145/2049706.2049708. Available at `https://arxiv.org/abs/0906.4586`.

A Correctness of rewrite rules

In this appendix we prove correctness for the rewrite rules of fig. 2.

Proof of proposition 3.1. We verify each rewrite rule by direct calculation. Recall that by lemma 2.5, for any Boolean-valued polynomial Q, $\overline{Q}(\mathbf{x},\mathbf{y}) = Q(\mathbf{x},\mathbf{y}) \mod 2$.

$$\text{[Elim]:}\quad \frac{1}{\sqrt{2^{m+2}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle = \frac{1}{\sqrt{2^{m+2}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m}(1+1)e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \frac{1}{\sqrt{2^m}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$

$$\text{[}\omega\text{]:}\quad \frac{1}{\sqrt{2^{m+1}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{4}y_0+\frac{1}{2}y_0Q(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m}\left(1+e^{2\pi i\left(\frac{1}{4}+\frac{1}{2}Q(\mathbf{x},\mathbf{y})\right)}\right)e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \begin{cases} \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m}(1+i)\,e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle & \text{if } Q(\mathbf{x},\mathbf{y}) = 0 \mod 2\\ \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m}(1-i)\,e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle & \text{if } Q(\mathbf{x},\mathbf{y}) = 1 \mod 2\end{cases}$$
$$= \begin{cases} \frac{1}{\sqrt{2^{m}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{8}+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle & \text{if } Q(\mathbf{x},\mathbf{y}) = 0 \mod 2\\ \frac{1}{\sqrt{2^{m}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{8}+\frac{3}{4}+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle & \text{if } Q(\mathbf{x},\mathbf{y}) = 1 \mod 2\end{cases}$$
$$= \frac{1}{\sqrt{2^m}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{8}+\frac{3}{4}\overline{Q}(\mathbf{x},\mathbf{y})+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle$$

$$\text{[HH]:}\quad \frac{1}{\sqrt{2^{m+1}}}\sum_{y_0\in\mathbb{Z}_2}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(\frac{1}{2}y_0(y_i+Q(\mathbf{x},\mathbf{y}))+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m}\left(1+e^{2\pi i(y_i+Q(\mathbf{x},\mathbf{y}))}\right)e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m,\, y_i=Q(\mathbf{x},\mathbf{y}) \mod 2}\left(1+e^{2\pi i(2k)}\right)e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$
$$+ \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m,\, y_i=1+Q(\mathbf{x},\mathbf{y}) \mod 2}\left(1+e^{2\pi i(2k+1)}\right)e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \frac{2}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m,\, y_i=Q(\mathbf{x},\mathbf{y}) \mod 2} e^{2\pi i R(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle$$
$$= \frac{1}{\sqrt{2^{m+1}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^m} e^{2\pi i\left(R[y_i\leftarrow\overline{Q}]\right)(\mathbf{x},\mathbf{y})}|\left(f[y_i\leftarrow Q]\right)(\mathbf{x},\mathbf{y})\rangle$$

[Case]: Recall the precondition

$$P(\mathbf{x},\mathbf{y}) = \frac{1}{4}y_i x + \frac{1}{2}y_i(y_j+Q(\mathbf{x},\mathbf{y})) + R(\mathbf{x},\mathbf{y}) = \frac{1}{4}y_j(1-x) + \frac{1}{2}y_j(y_i+Q'(\mathbf{x},\mathbf{y})) + R'(\mathbf{x},\mathbf{y}).$$

$$
\begin{aligned}
&\frac{1}{\sqrt{2^{m+2}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m+2}} e^{2\pi i P(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle \\
&\quad= \begin{cases} \frac{1}{\sqrt{2^{m+2}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m+2}} e^{2\pi i\left(\frac{1}{2}y_i(y_j+Q(\mathbf{x},\mathbf{y}))+R(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle & \text{if } x=0 \\ \frac{1}{\sqrt{2^{m+2}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m+2}} e^{2\pi i\left(\frac{1}{2}y_j(y_i+Q'(\mathbf{x},\mathbf{y}))+R'(\mathbf{x},\mathbf{y})\right)}|f(\mathbf{x},\mathbf{y})\rangle & \text{if } x=1 \end{cases} \\
&\quad= \begin{cases} \frac{1}{\sqrt{2^{m}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m}} e^{2\pi i\left(R[y_j\leftarrow\overline{Q}]\right)(\mathbf{x},\mathbf{y})}|\left(f[y_j\leftarrow Q]\right)(\mathbf{x},\mathbf{y})\rangle & \text{if } x=0 \\ \frac{1}{\sqrt{2^{m}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m}} e^{2\pi i\left(R'[y_i\leftarrow\overline{Q'}]\right)(\mathbf{x},\mathbf{y})}|\left(f[y_i\leftarrow Q']\right)(\mathbf{x},\mathbf{y})\rangle & \text{if } x=1 \end{cases} && \text{by [HH] and [Elim]} \\
&\quad= \frac{1}{\sqrt{2^{m}}}\sum_{\mathbf{y}\in\mathbb{Z}_2^{m}} e^{2\pi i\left((1-x)R[y_j\leftarrow\overline{Q}]+xR'[y_i\leftarrow\overline{Q'}]\right)(\mathbf{x},\mathbf{y})}|f(\mathbf{x},\mathbf{y})\rangle && \text{since } y_i,y_j\notin f
\end{aligned}
$$

□

B Reduction examples

In this appendix we give further examples of the use of our reduction rules to prove circuit identities.

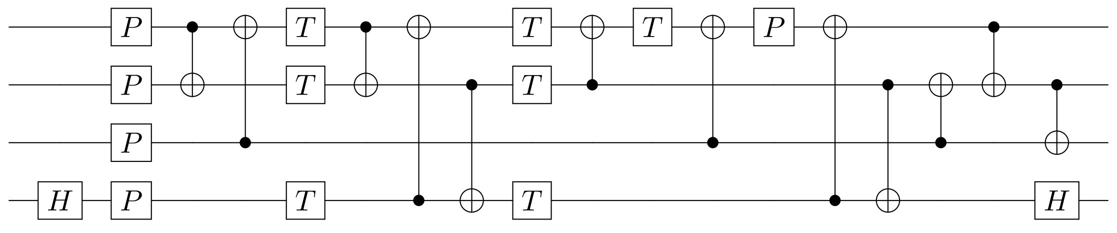
Example B.1. To show the use of the [ω] rule, we reduce the circuit $(SH)^3$ to the ω constant.

$$
\begin{aligned}
(\mathsf{SH})^3 : |x\rangle &\mapsto \frac{1}{\sqrt{2}^3}\sum_{y_1,y_2,y_3\in\mathbb{Z}_2} e^{2\pi i\frac{1}{8}(4xy_1+6y_1+4y_1y_2+6y_2+4y_2y_3+6y_3+1)}|y_3\rangle \\
&\mapsto \frac{1}{\sqrt{2}^3}\sum_{y_1,y_2,y_3\in\mathbb{Z}_2} e^{2\pi i\left(\frac{1}{2}(\frac{1}{2}y_1+y_1(y_2\oplus 1\oplus x))+\frac{1}{8}(6y_2+4y_2y_3+6y_3+1)\right)}|y_3\rangle \\
&\mapsto \frac{1}{\sqrt{2}^2}\sum_{y_2,y_3\in\mathbb{Z}_2} e^{2\pi i\frac{1}{8}(1-2(y2+1+x-2y_2-2x-2y_2x+4y_2x)+6y_2+4y_2y_3+6y_3+1)}|y_3\rangle && [\omega] \\
&\mapsto \frac{1}{\sqrt{2}^2}\sum_{y_2,y_3\in\mathbb{Z}_2} e^{2\pi i\frac{1}{8}(2x+4y_2x+4y_2y_3+6y_3)}|y_3\rangle \\
&\mapsto e^{2\pi i\frac{1}{8}(2x+6x)}|x\rangle && [\mathsf{HH},\ \mathsf{Elim}] \\
&\mapsto \omega|x\rangle.
\end{aligned}
$$

Example B.2. The one-bit full adder has the reversible path-sum specification

$$|x_1x_2x_3x_4\rangle \mapsto |x_1(x_1\oplus x_2)(x_1\oplus x_2\oplus x_3)(x_1x_2\oplus x_1x_3\oplus x_2x_3\oplus x_4)\rangle.$$

The implementation below over Clifford+T was obtained by using the Reed-Muller decoding method of [4] to reduce the number of T gates from the standard implementation using two Toffoli gates.



We can verify that this circuit implements the one-bit adder specification as follows:

$$\begin{aligned}
|x_1x_2x_3x_4\rangle &\mapsto \frac{1}{\sqrt{2}^2}\sum_{y_1,y_2\in\mathbb{Z}_2} e^{2\pi i\frac{1}{2}(y_1y_2+y_1x_1x_2+y_1x_1x_3+y_1x_2x_3+y_1x_4)}|x_1(x_1\oplus x_2)(x_1\oplus x_2\oplus x_3)y_2\rangle \\
&\mapsto \frac{1}{\sqrt{2}^2}\sum_{y_1,y_2\in\mathbb{Z}_2} e^{2\pi i\frac{1}{2}y_1(y_2+x_1x_2+x_1x_3+x_2x_3+x_4)}|x_1(x_1\oplus x_2)(x_1\oplus x_2\oplus x_3)y_2\rangle \\
&\mapsto |x_1(x_1\oplus x_2)(x_1\oplus x_2\oplus x_3)(x_1x_2\oplus x_1x_3\oplus x_2x_3\oplus x_4)\rangle && \textsf{[HH, Elim]}
\end{aligned}$$