How Many Quantum Circuit Identities Are Needed to Generate All Others?

Anonymous Authors

Anonymous Institution

Abstract. Quantum compiler optimization relies on rewriting rules derived from equivalences between quantum circuits, yet prior work has identified thousands of such identities, creating substantial challenges for their storage, management, and effective application. For many widely used unitary gate sets, including Clifford+T, this apparent complexity is largely redundant, raising a fundamental question: *How many quantum circuit identities are actually needed to generate all others?* In this work, we provide strong evidence that a small set of independent identities suffices to generate all circuit equivalences of bounded depth. Surprisingly, for circuits on up to nine qubits in which each side of an equality has depth at most ten, fewer than twenty independent identities are sufficient to derive all others, and for circuits on up to five qubits with depth at most ten, only 17 rules–each involving at most three qubits–are enough. These results enable significantly more compact and efficient rewriting systems for quantum compiler optimization and reveal underlying algebraic structure in common gate sets, showing that the vast majority of known circuit identities are consequences of a small foundational basis.

Keywords: Quantum Computing · Rewriting · Equational Theory.

1 Introduction

Quantum circuit optimization is essential for improving the performance and reliability of near-term devices, as reducing circuit size directly increases execution fidelity. Traditional compilers rely on manually designed rewrite rules and fixed optimization schedules, whereas recent automated approaches enumerate candidate circuits and verify equivalences to synthesize rewrite rules. Superoptimizers such as Quartz generate transformations via exhaustive enumeration and search-based rewriting [23], while other methods synthesize large collections of symbolic rules and apply probabilistic verification to produce thousands of rules–or even complete optimizers–for specific architectures in short time [22]. These advances demonstrate the power of automation, but also expose fundamental limitations in scalability and interpretability.

Despite this progress, two critical gaps remain. First, synthesis-based methods produce large and highly redundant rule sets whose application requires costly pattern matching over circuit graphs and scales poorly in practice; moreover, the practical value of most discovered rules–their frequency of use, generality across benchmarks, and mutual interactions–remains largely unexplored. Second, existing strategies treat rule discovery primarily as a combinatorial search problem, driven by enumeration or

randomized sampling, without exploiting the algebraic structure underlying quantum circuit identities. As a result, exploration is inefficient and provides little insight into why so many rules arise or how they relate to one another.

Classical Boolean circuits offer a striking contrast. Circuits over AND, OR, and NOT admit a compact and transparent equational theory in which all rewrites follow from a small set of fundamental laws such as commutativity, associativity, distributivity, and De Morgan's rules. Quantum circuits, by contrast, appear far more complex due to superposition, entanglement, and phase interactions. Indeed, prior work reports over 3,000 rewrite rules even for seven-qubit circuits, suggesting that quantum circuit identities form a fragmented and opaque collection rather than a coherent algebraic system. Moreover, composing even two simple equalities can generate seemingly new identities, revealing a substantial degree of redundancy and an apparent combinatorial explosion in the space of circuit equivalences.

Together, these observations raise a fundamental question.

How many quantum circuit identities are actually needed to generate all others?

Progress has been made in equational theories for quantum circuits [7,6], establishing minimal and complete axiom systems and showing that completeness in unrestricted settings typically requires infinitely many rules. Although mathematically elegant, these infinitely many rules offer limited insight for realistic bounded regimes. It remains unclear which equalities are necessary and sufficient and whether a small, manageable rule set can capture most circuit behavior in practice.

We adopt a complementary empirical perspective, asking whether a small set of natural identities suffices to capture the circuit equalities that matter in practice. To this end, we (i) analyze outputs of modern synthesis tools to expose systematic redundancy among discovered rules, and (ii) identify a small set of natural algebraic identities that form the basis of compact equational theories. In summary, this paper makes the following contributions:

- We uncover small, finite, and near-complete equational theories for several common quantum gate sets, bridging the gap between infinite but impractical axioms and large, redundant rule sets from synthesis tools.
- We demonstrate that these theories are relatively complete, capturing all circuit equalities up to the relevant size bound and reproducing all rewrite rules generated by existing synthesizers.
- We show that the resulting theories are concise, structured, and human-interpretable, providing both theoretical and practical insights into the minimal algebraic structure of quantum circuit optimization.

Together, these results suggest that equality-theory-driven synthesis and optimization provides a parsimonious and interpretable alternative to blind enumeration, while building a production-quality optimizer on top of such finite theories remains future work. Additionally, our generated rules provide principled guidance and empirical guarantees for designing effective rewriting strategies, capturing the most frequently applicable transformations in quantum circuit optimizations.

As a concrete illustration, Figure 1 presents a generated equational theory for Clifford+T circuits consisting of only 17 rules that derive all five-qubit identities up to

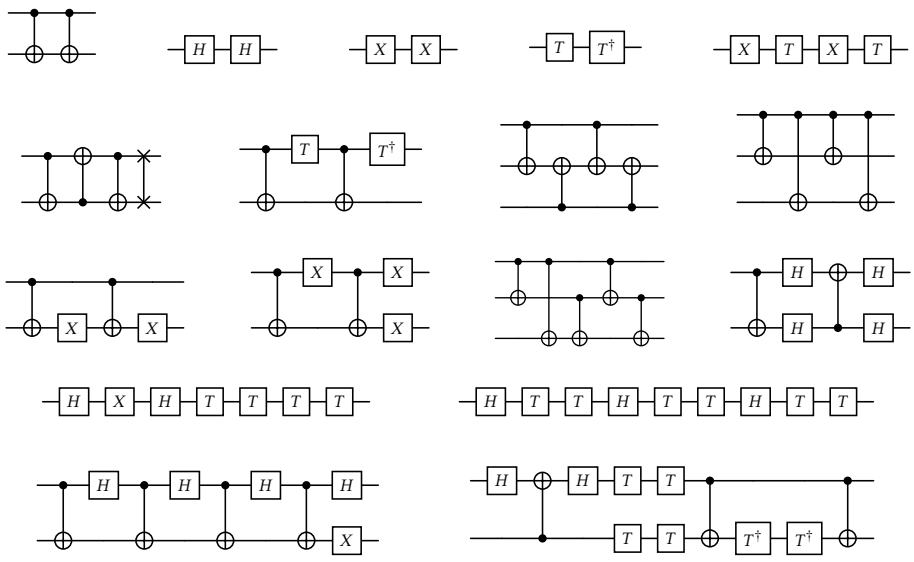


Fig. 1: An equational theory for Clifford+T circuits consisting of 17 rules that can derive all 5-qubit identities up to 16 gates. We simply use a quantum circuit C to represent an identity $C = I$ in the equational theory.

16 gates, with the largest rule containing 11 gates. Remarkably, each rule involves at most three qubits, yet together they suffice to generate all five-qubit circuit equalities. In other words, the seemingly complex space of five-qubit identities used for circuit optimization can be fully captured by only three-qubit equalities, revealing an unexpectedly compact and structured underlying system. In contrast to prior approaches that either yield infinite but impractical axiom systems [7] or thousands of highly redundant rules [22], our method discovers small, finite theories that are nearly complete and practically useful. Moreover, the resulting rules are concise, structured, and human-interpretable, revealing unexpected simplicity in common quantum gate sets. For example, two rules in Figure 1 can be written as

$$CZ \cdot CX = (X \otimes I) \cdot CX \cdot CZ, \qquad S^\dagger \otimes I = CX \cdot CY \cdot CZ.$$

These transparent identities not only facilitate efficient reasoning about circuit equivalence but also illuminate the underlying algebraic structure of quantum circuits.

The remainder of the paper is organized as follows. Section 2 introduces the necessary preliminaries. Section 3 presents our synthesis algorithm. Section 4 describes our method for pruning and minimizing equational theories. Section 5 evaluates our approach on multiple gate sets. Section 6 reviews related work, and Section 7 concludes. We defer most proofs to the appendix.

2 Preliminaries

2.1 Quantum States and Operations

A *quantum state* describes the physical state of a quantum system. A single-qubit pure state $|\psi\rangle$ is a unit vector in a two-dimensional Hilbert space and may be written as a superposition of $|0\rangle$ and $|1\rangle$. An n-qubit system is represented in a 2^n-dimensional Hilbert space, enabling superposition and entanglement.

Quantum operations are modeled as unitary transformations on this Hilbert space. A unitary matrix U satisfies $U^\dagger U = I$ and preserves state norms. It acts on pure and mixed states as $|\psi\rangle \mapsto U|\psi\rangle$ and $\rho \mapsto U\rho U^\dagger$, respectively. Common single-qubit gates include H, T, and the Pauli gates $\{I, X, Y, Z\}$, while standard two-qubit gates include SWAP and CNOT.

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

2.2 Quantum Circuits

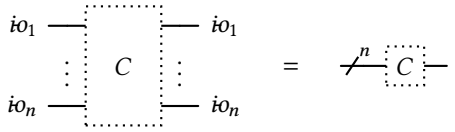
Quantum circuits implement quantum computations by composing quantum gates. A circuit is a sequence of gates acting on qubits, typically depicted as wires connected by gate symbols, and its semantics is given by the composition of the corresponding unitary operations. In this work, we adopt the port graph formalism to obtain a minimal and canonical representation of quantum circuits. Under this formalism, a circuit is modeled as a directed graph whose vertices represent quantum gates and whose edges represent qubit wires. Each vertex is equipped with a set of *ports* that serve as connection points for edges and correspond to the qubits on which the gate operates. To capture the reversibility of quantum operations, each port functions simultaneously as an input and an output. We formalize this representation as follows.

Definition 1 (Quantum Circuit). A quantum circuit over a quantum gate set $\mathcal{G}$ is a tuple $C = (V, g, m)$. Specifically,

- V is a finite set of vertices (nodes), each representing a quantum gate.
- $g : V \to \mathcal{G}$ labels each vertex with a gate from $\mathcal{G}$.
- $m : \text{Ports}(C) \leftrightarrow \text{Ports}(C)$ is a bijection mapping (perfect matching) that connect each port to its associated port.
- The set of all ports $\text{Ports}(C)$ is defined as $\{(v, k) \mid v \in V, 1 \le k \le g(v).\text{arity}\} \cup \{(io, k) \mid 1 \le k \le n\}$ where the first part represents all the ports associated with the gates and in the second part, we use special label io to denote the input/output qubit ports of the circuit. Each port is identified by a pair consisting of a vertex (or io) and a port index.

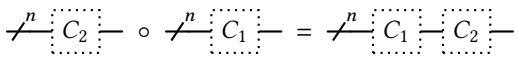
- m connects ports in forward direction, and the inverse mapping m^{-1}, connect ports in backward direction. For example, m will map $(\text{i}\!\text{o}, k)$ ports to the initial gate applied on qubit k, and m^{-1} will map $(\text{i}\!\text{o}, k)$ to the final gate applied on k.
- For the well-formedness of quantum circuits, we require that the corresponding directed graph (V, E) is acyclic, where $E = \{(v_1, v_2) \mid (v_1, k_1) \in P, m(v_1, k_1) = (v_2, k_2)\}$ is the set of directed edges generated by ports. This guarantees there exists a topological order of executing gates in V, so that no gate will be applied before its input qubits are ready.

For convenience, we call $|C| = |V|$ the size of the quantum circuit. Similarly, we denote the set of all n-qubit quantum circuits over gate set $\mathcal{G}$ as $\text{Circ}_n(\mathcal{G})$. We specify the size using superscripts: $\text{Circ}_n^k(\mathcal{G})$ for exactly k gates, and $\text{Circ}_n^{\leq k}(\mathcal{G})$ for at most k gates. We also use diagrammatic representation for quantum circuits, where we represent the circuit C using a conventional quantum circuit diagram with n wires, as shown below, where each gate represents a vertex, each line represents a connection between ports, and the left/right endpoints represents input/output ports:

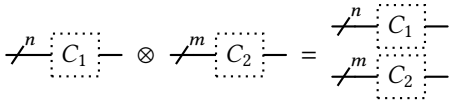


The reason of using this representation of circuits is its topological treatment of permutation gates: rather than appearing as distinct nodes, these operations are implicitly encoded through the permutation of edges. Consequently, this yields a canonical form where the underlying graph structure remains invariant regardless of the explicit placement or arrangement of permutation gates. To show this, we define the following operations for quantum circuits.

- Composition $\circ : \text{Circ}_n(\mathcal{G}) \times \text{Circ}_n(\mathcal{G}) \to \text{Circ}_n(\mathcal{G})$ is graphically depicted as



- Tensor product $\otimes : \text{Circ}_n(\mathcal{G}) \times \text{Circ}_m(\mathcal{G}) \to \text{Circ}_{n+m}(\mathcal{G})$ is depicted as



- Identity circuits $id_n : C_n(\mathcal{G})$ which is graphically identical to $\not{}^{n}$.
- Permutation circuits $\sigma^{\text{c}} : \text{Circ}_n(\mathcal{G})$ which represent the permutation of qubits according to the symmetry $\sigma \in \text{Sym}(n)$, formally defined as $\sigma^{\text{c}} = (\varnothing, \varnothing, m)$ where $m(\text{i}\!\text{o}, k) = (\text{i}\!\text{o}, \sigma(k))$ for $1 \leq k \leq n$.
- Adjoint circuits $(\cdot)^\dagger : \text{Circ}_n(\mathcal{G}) \to \text{Circ}_n(\mathcal{G})$ which is defined as $(V, g, m)^\dagger = (V, g^\dagger, m^{-1})$ where $g^\dagger(v) = g(v)^\dagger$ for $v \in V$.

– Generator circuits $G(k_1, \ldots, k_m) : \mathrm{Circ}_n^1(\mathcal{G})$ which represents the application of gate $G \in \mathcal{G}$ on qubits $k_1, \ldots, k_m$ (where $m = G.\mathrm{arity}$), formally defined as $G(k_1, \ldots, k_m) = (\{v\}, \lambda v.G, m)$ where

$$\begin{aligned} m(\dot{\imath}o, k_i) = (v, i) \quad m(v, i) = (\dot{\imath}o, k_i) \quad & \text{for } 1 \le i \le m, \\ m(\dot{\imath}o, j) = (\dot{\imath}o, j) \quad & \text{for } j \notin \{k_1, \ldots, k_m\}. \end{aligned} \tag{2.1}$$

We denote the set of all generator circuits as $\mathrm{Gen}_n(\mathcal{G})$, note that $\mathrm{Gen}_n(\mathcal{G}) \subsetneq \mathrm{Circ}_n^1(\mathcal{G})$ for $n > 1$ because additional permutation can be embedded in $\mathrm{Circ}_n^1(\mathcal{G})$.

These operations induce several basic properties of our circuit representation.

1. Our representation of quantum circuits remain the same regardless of different composition of permutations gates, which can be easily observed from
$$(\sigma_2 \circ \sigma_1)^{\mathrm{c}} = \sigma_2^{\mathrm{c}} \circ \sigma_1^{\mathrm{c}} \text{ and } (\sigma^{-1})^{\mathrm{c}} = (\sigma^{\mathrm{c}})^{\dagger}. \tag{2.2}$$
2. Generator circuits can pass through σ^c by permuting the qubits it acts on, i.e.,
$$\sigma^{\mathrm{c}} \circ G(k_1, \ldots, k_m) = G(\sigma(k_1), \ldots, \sigma(k_m)) \circ \sigma^{\mathrm{c}}. \tag{2.3}$$
3. Finally, $\sigma^{\mathrm{c}} \circ C \circ (\sigma^{-1})^{\mathrm{c}}$ forms a circuit of C with only the qubit wires permuted in the circuit diagram, without really adding any additional permutation gates in the circuit diagram.

These properties imply that our representation of quantum circuits effectively captures the essential structure of quantum operations while abstracting away the superficial arrangement of permutation gates. This canonical form simplifies the analysis and manipulation of quantum circuits, particularly in the context of equational reasoning. Now let us define the equational logic for quantum circuits.

2.3 Quantum Circuit Equational Logic

In this section, we establish the foundations of equational logic and equational theory as applied to quantum circuits. We begin by defining the syntax for quantum circuit equations, which serves as the basis for our equational proof system.

Definition 2 (QCEL-Equation). A quantum circuit equation over a quantum gate set $\mathcal{G}$ is an n-qubit quantum circuit $C \in \mathrm{Circ}_n(\mathcal{G})$ in the canonical form $C \approx id_n$. For convenience, we also write $C_1 \approx C_2$ as a shorthand for $C_1 \circ C_2^{\dagger} \approx id_n$ where $C_1, C_2 \in \mathrm{Circ}_n(\mathcal{G})$.

With this notation, we then define the equational proof system for quantum circuits via the inference rules presented in Figure 2.

Definition 3 (QCEL-Derivability). Let Γ be a set of quantum circuit equations over a quantum gate set $\mathcal{G}$, and let C be an n-qubit quantum circuit. The judgment

$$\Gamma \vdash C \approx id_n$$

signifies that $C \approx id_n$ is derivable from Γ using the inference rules in Figure 2.

$$\textsc{Eq}\ \frac{}{\Gamma \vdash id_n \approx id_n} \qquad \textsc{Assume}\ \frac{C \approx id_n \in \Gamma}{\Gamma \vdash C \approx id_n} \qquad \textsc{Adjoint}\ \frac{\Gamma \vdash C \approx id_n}{\Gamma \vdash C^\dagger \approx id_n} \qquad \textsc{Extend}\ \frac{\Gamma \vdash C \otimes id_m \approx id_{n+m}}{\Gamma \vdash C \otimes id_k \approx id_{n+k}}$$

$$\textsc{Rotate}\ \frac{\Gamma \vdash C_1 \circ C_2 \approx id_n}{\Gamma \vdash C_2 \circ C_1 \approx id_n} \qquad \textsc{Permutation}\ \frac{\Gamma \vdash C \approx id_n \qquad \sigma \in \mathrm{Sym}(n)}{\Gamma \vdash \sigma^{\mathrm{c}} \circ C \circ (\sigma^{\mathrm{c}})^\dagger \approx id_n} \qquad \textsc{Transitivity}\ \frac{\Gamma \vdash C_1 \approx C_2 \qquad \Gamma \vdash C_2 \approx C_3}{\Gamma \vdash C_1 \approx C_3}$$

Fig. 2: Equational Proof Rules for Quantum Circuits

To provide intuition for the inference rules in Figure 2:

- The Rotate rule permits the cyclic rotation of circuits within an equation. Physically, this holds because $[\![C_1]\!][\![C_2]\!] = I$ implies $[\![C_2]\!] = [\![C_1]\!]^\dagger$, which consequently implies $[\![C_2]\!][\![C_1]\!] = I$.
- The Permutation rule exploits the symmetry of qubits. If a circuit C reduces to the equation, any permutation of the wires in C (conjugated by that permutation) remains an equation operation.
- The Transitivity rule enables the chaining of equations and circuit substitution. For instance, $C_1 \circ C_2 \circ C_3 \approx id_n$ is equivalent to $C_2 \approx C_1^\dagger \circ C_3^\dagger$. Thus, if $C_2 \approx C_4$, we may deduce $C_4 \approx C_1^\dagger \circ C_3^\dagger$, or equivalently $C_1 \circ C_4 \circ C_3 \approx id_n$.

With this in mind, we can now define the semantics for our notation of quantum circuit equations. Since our paper doesn't involve symbolic circuits, we directly define the semantics using the quantum gates' unitary matrix interpretation.

Definition 4 (QCEL-Identity). Let $C \approx id_n$ be an n-qubit quantum circuit equation over a quantum gate set $\mathcal{G}$, we call $C \approx id_n$ an *identity* if it holds that:

$$\models C \approx id_n \iff [\![C]\!] = e^{\mathrm{i}\alpha} I^{\otimes n}$$

where $[\![C]\!]$ is the unitary matrix on $\mathrm{U}(2^n)$ derived using the quantum gate set $\mathcal{G}$, and $\alpha \in \mathbb{R}$ represents a global phase that may arise during the computation.

Similar to circuits, we denote the set of n-qubit quantum circuit identities over $\mathcal{G}$ as $\mathrm{Id}_n(\mathcal{G})$, and use $\mathrm{Id}_n^k(\mathcal{G})$ and $\mathrm{Id}_n^{\leq k}(\mathcal{G})$ to specify equations with exactly k gates and at most k gates, respectively. We also call an equation set Γ as an *equational theory* if all equations in Γ are identities, denoted as $\models \Gamma$.

In this paper, we adopt a simple definition of the soundness and completeness of the equational logic, defined as follows.

Theorem 1 (QCEL Soundness). For any equational theory Γ over a quantum gate set $\mathcal{G}$ and any n-qubit quantum circuit C, if $\Gamma \vdash C \approx id_n$ and $\models \Gamma$, then $\models C \approx id_n$.

Definition 5 (QCEL Relative Completeness). An equational theory Γ over gate set $\mathcal{G}$ is complete with respect to an equation set $\mathcal{I}$ if, for any equation $C \approx id_n \in \mathcal{I}$, $\models C \approx id_n \Leftrightarrow \Gamma \vdash C \approx id_n$.

3 Identity Synthesis Algorithm

We present an algorithm to synthesize all quantum circuit identities for a given gate set and size bound. To evaluate empirical completeness, we systematically enumerate circuits and test equivalence using canonical representatives, following [3]. In contrast to prior work limited to Clifford circuits, our approach extends to universal gate sets by relying only on equivalence classes generated by permutation gates.

3.1 Overview

First we define our representative reduce function $\textsc{Reduce} : \mathrm{U}(2^n) \to \mathrm{U}(2^n)$ that maps a quantum operator U to its canonical representative in the equivalence class $\left\{ [\![\sigma_2^{\mathrm{c}}]\!]\, U\, [\![\sigma_1^{\mathrm{c}}]\!] \;\middle|\; \sigma_1, \sigma_2 \in \mathrm{Sym}(n) \right\}$. In other words, $\textsc{Reduce}$ satisfies the following:

$$\textsc{Reduce}(U_1) = \textsc{Reduce}(U_2) \iff \exists \sigma_1, \sigma_2 \in \mathrm{Sym}(n) \text{ s.t. } U_1 = [\![\sigma_2^{\mathrm{c}}]\!]\, U_2\, [\![\sigma_1^{\mathrm{c}}]\!] \tag{3.1}$$

A specific implementation of the function of $\textsc{Reduce}$ is described in Section 3.2. Now let us define the result of the algorithm. Given a quantum gate set $\mathcal{G}$, a circuit size k, and number of qubits n, our synthesis algorithm aims to compute $\mathcal{R}_n^k[\mathcal{G}] : \mathrm{U}(2^n) \to \mathcal{P}(\mathrm{Circ}_n^k(\mathcal{G}))$ defined as follows:

$$\mathcal{R}_n^k[\mathcal{G}](U) = \left\{ C \;\middle|\; [\![C]\!] = \textsc{Reduce}([\![C]\!]) = U,\ C \in \mathrm{Circ}_n^k(\mathcal{G}) \right\} \tag{3.2}$$

We also write $\mathcal{R}_n^{\leq k}[\mathcal{G}](U) = \bigcup_{1 \leq i \leq k} \mathcal{R}_n^i[\mathcal{G}](U)$ similar to circuits and identities. Intuitively, $\mathcal{R}_n^k[\mathcal{G}]$ contains all circuits of size k over n qubits evaluate to the canonical representative U. Note that $\mathcal{R}_n^k[\mathcal{G}]$ provides a compact representation of all pair of equivalent circuits in $\mathrm{Circ}_n^k(\mathcal{G})$ and their unitaries, according to the following lemma:

Lemma 1. $\mathrm{Circ}_n^k(\mathcal{G}) = \left\{ \sigma_2^{\mathrm{c}} \circ C \circ \sigma_1^{\mathrm{c}} \;\middle|\; C \in \mathcal{R}_n^k[\mathcal{G}](U),\ U \in \mathrm{U}(2^n),\ \sigma_1, \sigma_2 \in \mathrm{Sym}(n) \right\}$.

Proof. ($\supseteq$) Trivial from the definition of $\mathrm{Circ}_n^k(\mathcal{G})$. ($\subseteq$) For any circuit $C' \in \mathrm{Circ}_n^k(\mathcal{G})$, let $U = \textsc{Reduce}([\![C']\!])$. By the definition of $\textsc{Reduce}$, there exist $\sigma_1, \sigma_2 \in \mathrm{Sym}(n)$ such that $U = [\![\sigma_2^{\mathrm{c}}]\!]\, [\![C']\!]\, [\![\sigma_1^{\mathrm{c}}]\!]$. Let $C = \sigma_2^{\mathrm{c}} \circ C' \circ \sigma_1^{\mathrm{c}}$, then $[\![C]\!] = U$ and $C \in \mathcal{R}_n^k[\mathcal{G}](U)$. Therefore, $C' \in \mathcal{R}_n^k[\mathcal{G}]$. □

Then we also need to prove that $\mathcal{R}_n^k[\mathcal{G}]$ can be used to naturally converted to a set of quantum circuit identities by combining every two circuits that evaluate to the same unitary. Specifically, for each unitary U such that $\mathcal{R}_n^k[\mathcal{G}](U) \neq \emptyset$, we can generate the identities $C_1 \approx C_2$ for all distinct pairs $C_1, C_2 \in \mathcal{R}_n^{\leq k}[\mathcal{G}](U)$. The following lemma states that this conversion yields all circuit identities of size up to $2k$.

Lemma 2. $\mathrm{Id}_n^{\leq 2k}(\mathcal{G}) = \left\{ \sigma^{\mathrm{c}} \circ C_1 \circ C_2^{\dagger} \circ (\sigma^{-1})^{\mathrm{c}} \approx id_n \;\middle|\; C_1, C_2 \in \mathcal{R}_n^{\leq k}[\mathcal{G}](U),\ \sigma \in \mathrm{Sym}(n) \right\}$.

Algorithm 1: Quantum Circuit Synthesis Algorithm with Reduce

Input : Quantum gate set $\mathcal{G}$, Maximum circuit size k, and number of qubits n.
Output : A mapping $\mathcal{R}_n^{\leq k}[\mathcal{G}] : \mathrm{U}(2^n) \to \mathcal{P}(\mathrm{Circ}_n^{\leq k}(\mathcal{G}))$.

fn Synth$(\mathcal{G}, k, n)$ **:**
  $R^1 = \mathcal{R}_n^1[\mathcal{G}]$
  for $i \leftarrow 2$ **to** k **:**
    invariant $R^{i-1} = \mathcal{R}_n^{i-1}[\mathcal{G}]$ // $\mathcal{R}_n^k[\mathcal{G}]$ (3.2), C.f. Theorem 2
    Initialize $R^i = \lambda U.\varnothing$
    for $U \in \mathrm{U}(2^n)$ that $R^i(U) \neq \varnothing$ **:**
      for $G \in \mathrm{Gen}_n(\mathcal{G})$ **:** // $\mathrm{Gen}_n(\mathcal{G})$ (2.1)
        $U' = \textsc{Reduce}(\llbracket G \rrbracket U)$ // See Section 3.2
        for $\sigma_1, \sigma_2 \in \mathrm{Sym}(n)$ that $\exists \alpha.\ U' = e^{\mathrm{i}\alpha} \llbracket \sigma_2^{\mathrm{c}} \rrbracket \llbracket G \rrbracket U \llbracket \sigma_1^{\mathrm{c}} \rrbracket$ **:**
          $R^i(U') = R^i(U') \cup \{\sigma_2^{\mathrm{c}} \circ G \circ C \circ \sigma_1^{\mathrm{c}} \mid C \in R^{i-1}(U)\}$
  return $R^1 \cup R^2 \cup \ldots \cup R^k$

Using these notation, we can now describe our synthesis algorithm as shown in Algorithm 1. The algorithm iteratively constructs the mappings $\mathcal{R}_n^1[\mathcal{G}], \mathcal{R}_n^2[\mathcal{G}], \ldots, \mathcal{R}_n^k[\mathcal{G}]$. In each iteration i, it initializes an empty mapping R^i and populates it by extending circuits from the previous iteration R^{i-1} with one additional gate from the gate set $\mathcal{G}$. For each unitary U in the previous mapping R^{i-1}, it considers all possible single-gate extensions G and computes the new unitary $U' = \textsc{Reduce}(\llbracket G \rrbracket U)$. It then finds all permutations σ_1, σ_2 such that U' can be expressed as $\llbracket \sigma_2^{\mathrm{c}} \rrbracket \llbracket G \rrbracket U \llbracket \sigma_1^{\mathrm{c}} \rrbracket$. Finally, it updates the mapping R^i by adding the new circuits formed by $\sigma_2^{\mathrm{c}} \circ G \circ C \circ \sigma_1^{\mathrm{c}}$ for each circuit C in $R^{i-1}(U)$. The following theorem states the correctness of the algorithm.

Theorem 2. Algorithm 1 correctly computes the mapping $\mathcal{R}_n^{\leq k}[\mathcal{G}]$ for the given quantum gate set $\mathcal{G}$, circuit size k, and number of qubits n.

3.2 Implementation of Reduce

Now we describe our implementation of the Reduce function used in Algorithm 1. The goal of Reduce is to find the canonical representative of a unitary U under the equivalence class generated by two different permutation gates on both sides: $\left\{\llbracket \sigma_2^{\mathrm{c}} \rrbracket U \llbracket \sigma_1^{\mathrm{c}} \rrbracket \;\middle|\; \sigma_1, \sigma_2 \in \mathrm{Sym}(n)\right\}$. To achieve this, we first introduce the notion of *seemingly ordered quantum state* that helps us to reduce the search space of possible symmetries.

Definition 6 (Seemingly Ordered Quantum State). A n-qubit quantum state $|\psi\rangle$ is called to be *seemingly ordered* if for any two qubit $1 \leq i < j \leq n$, we have:

$$\left(\langle\psi|2^i\rangle, \langle\psi|2^n - 1 - 2^i\rangle\right) \sqsubseteq \left(\langle\psi|2^j\rangle, \langle\psi|2^n - 1 - 2^j\rangle\right)$$

where $\sqsubseteq$ is the lexicographical order on complex numbers defined as: for any two complex numbers $a = a_r + ia_i$ and $b = b_r + ib_i$, $a \sqsubseteq b$ if and only if either $a_r < b_r$, or $a_r = b_r$ and $a_i \leq b_i$. We denote the set of all symmetries $\sigma \in \mathrm{Sym}(n)$ that converts the quantum state $|\psi\rangle$ into a seemingly ordered state $\llbracket \sigma^{\mathrm{c}} \rrbracket |\psi\rangle$ as $\mathrm{Sym}^{\approx}(\psi)$.

Algorithm 2: Implementation of the REDUCE function

Parameters: ψ_l a randomized quantum state on symmetric qubits. ψ_r a fully randomized quantum state.
Input : A unitary U on n qubits.
Output : $U_{\min} = \textsc{Reduce}(U)$ and $\Sigma = \{(\sigma_l, \sigma_r) \mid \exists \alpha.\ \sigma_l U \sigma_r = e^{i\alpha} U_{\min}\}$.

fn REDUCEIMPL(U) **:**
  Initialize $\psi_{\min} = \top$, $U_{\min} = U$, $\Sigma = \varnothing$
  $\psi_0 = \text{deφase}\ U^\dagger|\psi_l\rangle$ // deφase (3.3), C.f. Sec 3.3 (5)
  foreach $\sigma_r \in \text{Sym}^{\approx}(\psi_0)$ **:** // $\text{Sym}^{\approx}$ Definition 6
    $|\psi\rangle = \text{deφase}\ U \llbracket \sigma_r^c \rrbracket^\dagger |\psi_r\rangle$
    assert $|\text{Sym}^{\approx}(\psi)| = 1$ // See Theorem 3
    $\{\sigma_l\} = \text{Sym}^{\approx}(\psi)$
    if $\llbracket \sigma_l^c \rrbracket \psi \sqsubset \psi_{\min}$ **:**
      $\Sigma = \varnothing;\ \psi_{\min} = \llbracket \sigma_l^c \rrbracket \psi;\ U_{\min} = \llbracket \sigma_l^c \rrbracket U \llbracket \sigma_r^c \rrbracket^\dagger$
    if $\llbracket \sigma_l^c \rrbracket \psi = \psi_{\min}$ **:**
      assume $U_{\min} = e^{i\alpha} \llbracket \sigma_l^c \rrbracket U \llbracket \sigma_r^c \rrbracket^\dagger$ for some $\alpha \in \mathbb{R}$ // See Theorem 4
      $\Sigma = \Sigma \cup \{(\sigma_l, \sigma_r^\dagger)\}$
  return $U_{\min}, \Sigma$

Algorithm 2 shows our implementation of the REDUCE function. The key idea is to apply U twice, once backward, to find possible right-side symmetries, and once forward to find the corresponding left-side symmetries. To do this, we need two initial quantum states: the *left-initial* state $|\psi_l\rangle$ is a randomized quantum state on symmetric qubits, i.e., $\llbracket \sigma^c \rrbracket |\psi_l\rangle = |\psi_l\rangle$ for any $\sigma \in \text{Sym}(n)$; and the *right-initial* state $|\psi_r\rangle$ is a fully randomized quantum state, i.e., $|\text{Sym}^{\approx}(\psi_r)| = 1$. We require the left-initial state to be symmetric so that after applying U from the backward, the resulting state $|\psi_0\rangle$ can help us to reduce the search space of possible right-side symmetries. The right-initial state is fully randomized to ensure that $|\psi\rangle$ has a unique left-side symmetry mapping it to a canonical form after applying U and a right-side symmetry.

In Algorithm 2, we first apply U from the backward. We compute $U^\dagger|\psi_l\rangle$ and then normalize the state using the following phase removing function:

$$\text{deφase}\ |\psi\rangle = \frac{|\langle\psi|0\rangle|}{\langle\psi|0\rangle}|\psi\rangle \tag{3.3}$$

This normalization ensures that the amplitude of the $|0\rangle$ basis state is always a non-negative real number, which helps to eliminate any redundant phase factor arisen from quantum circuits. Then, we obtain state $|\psi_0\rangle$, to be used to reduce search space of right-side symmetries. Next, we iterate through all symmetries $\sigma_r \in \text{Sym}^{\approx}(\psi_0)$. Each of such a symmetry σ_r represents a potential permutation on the right side of U.

For each such symmetry, we compute forward by applying $U \llbracket \sigma_r^c \rrbracket^\dagger$ to the right-initial state $|\psi_r\rangle$. This results in a new quantum state $|\psi\rangle$. Since $|\psi_l\rangle$ is fully-randomized and independent to $U \llbracket \sigma_r^c \rrbracket^\dagger$, very likely, $|\psi\rangle$ has a unique left-side symmetry that sorts it into a seemingly ordered state. We assume this because: 1) This assertion holds up to floating-point numerical precision, as shown in Theorem 3; 2) in our experiments, we

have never encountered a case where $|\mathrm{Sym}^{\approx}(\psi)| > 1$. Therefore, we directly extract the unique left-side symmetry σ_l from $\mathrm{Sym}^{\approx}(\psi)$.

We then compare the resulting state $[\![\sigma_l^c]\!]|\psi\rangle$ with the current minimum state $\psi_{\min}$ using the lexicographical order $\sqsubset$. If it is smaller, we update $\psi_{\min}$ and $U_{\min}$. If it is smaller or equal, record the corresponding symmetries (σ_l, σ_r). After enumerating all σ_r, we return the minimal unitary $U_{\min}$ and its symmetry set Σ. Σ will be used at line 9 in Algorithm 1 to enumerate all equivalent circuits of U.

Note that we assume at line 11 that if two states are equal, then their corresponding unitaries are also equal. In Theorem 4, we prove this observation only fails due to the inaccuracies of floating-point arithmetic. To verify the assumption at line 11 in Algorithm 2 practically holds, we observed this fact that if this assumption is incorrect, the incorrect Σ will cause line 9 in Algorithm 1 to introduce incorrect equivalences in $\mathcal{R}_n^k[\mathcal{G}]$. However, in our experiments reported in Section 5, especially the correctness validation in Section 5.2, we didn't find any incorrect identities synthesized by our algorithm. Thus, we are confident about the practical correctness of this assumption.

We then discuss the correctness of Algorithm 2. First, we justify the validity of the assertion at line 6 and the assumption at line 11. The following theorems hold almost surely under sufficient numerical precision:

Theorem 3. Given a unitary U on n qubits, and a fully randomized quantum state $|\psi_r\rangle$ independent to U, $\Pr\left(\left|\mathrm{Sym}^{\approx}\left(\text{deφase } U|\psi_r\rangle\right)\right| > 1\right) = 0$.

Intuitively, the condition of $\mathrm{Sym}^{\approx}$ is pretty hard for random $|\psi_r\rangle$ to satisfy.

Theorem 4. Given two unitaries U_1 and U_2, and a fully randomized quantum state $|\psi_r\rangle$ independent to both U_1 and U_2, if deφase $U_1|\psi_r\rangle$ = deφase $U_2|\psi_r\rangle$ holds, then the posterior probability $\Pr\left(\exists\alpha.\ U_1 = e^{\mathrm{i}\alpha}U_2\right) = 1$.

It is very hard for $|\psi_r\rangle$ to be an eigenvector of $U_2^\dagger U_1$.

Next, we need to prove that Algorithm 2 practically returns a canonical representative of U under the equivalence class generated by double permutation gates. This can be shown by the following theorem, which proves the correctness of Algorithm 2.

Theorem 5. Algorithm 2 returns $U_{\min}$ satisfying (3.1) under the assertion and assumption.

Theorem 6. If the assertion and assumption hold, Algorithm 2 returns

$$\Sigma = \{(\sigma_l, \sigma_r) \mid \exists\alpha,\ \sigma_l U \sigma_r = e^{\mathrm{i}\alpha}U_{\min}\}.$$

3.3 Implementation Details

We provide additional implementation notes for our synthesis algorithm here:

1. We implement our algorithm using floating-point arithmetic, which can introduce numerical errors. However, as demonstrated in Section 5.2, such numerical errors can be avoided by choosing sufficiently precise floating-point representations.

2. To minimize the execution time within Reduce, we don't construct the full unitary matrix U of the input circuit. Instead, we only simulate the action of U using a corresponding circuit C by applying its gates sequentially on the input state.
3. To save the space to store all U in $\mathcal{R}_n^k[\mathcal{G}]$, we only store the hash values of U using a high-quality hash function as keys in the mapping. In Section 5.2, we didn't find any hash collision that causes the equivalence classes of $\mathcal{R}_n^k[\mathcal{G}](U)$ not provable using our equational theory minimization procedure.
4. In Algorithm 1, maintaining a set for all circuits at line 10 is too costly. Instead, for each $\mathcal{R}_n^k[\mathcal{G}]$, we only keep track of each quadruple $(\sigma_1, G, U, \sigma_2)$. Similarly, we also filtered out all duplicate circuits in the equivalence class of each $\mathcal{R}_n^k[\mathcal{G}](U)$, that simply adds additional gates to existing equations, which simplifies our equational theory minimization process.
5. In the deφase function, the amplitude $\langle\psi|0\rangle$ may be zero during backward evaluation. When this occurs, we instead remove the phase of the sum of amplitudes $\sum_{i=0}^{2^n-1}\langle\psi|i\rangle$ from the state $|\psi\rangle$. This guarantees the normalization step is always symmetric to different order of qubits, i.e. deφase $[\![\sigma^c]\!]|\psi\rangle = [\![\sigma^c]\!] \cdot$ deφase $|\psi\rangle$. Therefore, the correctness of Reduce is preserved.

4 Equational Theory Pruning

In this section, we describe our algorithm for pruning a given equational theory of quantum circuits. The goal of the theory pruning is to find a small set of circuit identities that can derive all circuit identities in $\mathrm{Id}_n^{\le k}(\mathcal{G})$. Similar to prior work on circuit optimization [23], we utilized an algorithm similar to TASO [16], which iteratively applies rewrite rules to test if an identity can be derived from other identities.

Some rules like Permutation and Rotate in Definition 3 are easy to deduce using the following canonical representative: CanonC and CanonI. Intuitively, CanonC maps a circuit to its canonical representative in the equivalence class defined by the Permutation rules, while CanonI maps an identity to its canonical representative in the equivalence class defined by the Permutation and Rotate rules, which satisfy the following properties:

$$\textsc{CanonC}(C_1) = \textsc{CanonC}(C_2) \iff C_1 \approx \mathrm{id}_n \xleftrightarrow{\textsc{Permutation}} C_2 \approx \mathrm{id}_n \tag{4.1}$$

$$\textsc{CanonI}(I_1) = \textsc{CanonI}(I_2) \iff I_1 \xleftrightarrow[\textsc{Rotate}]{\textsc{Permutation}} I_2 \tag{4.2}$$

We implement these two canonical representative by finding the lexicographically smallest circuit and identity in their respective equivalence classes. This can be efficiently done by enumerating all permutations of qubits and rotations of circuits, which is feasible since the number of qubits n is small in our setting.

With these two canonical representative, we can now present our equational theory pruning algorithm in Algorithm 3. The algorithm takes as input the mapping $\mathcal{R}_n^{\le k}[\mathcal{G}]$ generated by our synthesis algorithm in Section 3 and outputs a small set of circuit identities that can derive all identities in $\mathrm{Id}_n^{\le 2k}(\mathcal{G})$. The main idea of the algorithm is to first convert $\mathcal{R}_n^{\le k}[\mathcal{G}]$ into a set of circuit identities $\mathcal{I}$, then iteratively try

to prove each identity in $\mathcal{I}$ using previously proved identities and a set of rewrite rules derived from $\mathcal{R}_n^{\leq k}[\mathcal{G}]$. If an identity cannot be proved within a certain search limit, it is added to the set of assumed identities, which will be included in the final output.

Algorithm 3: Equational Theory Pruning Algorithm

Input : $R = \mathcal{R}_n^{\leq k}[\mathcal{G}]$ generated by Algorithm 1.
Output : A small set of circuit identities that can derive all identities in $\mathrm{Id}_n^{\leq 2k}(\mathcal{G})$.

fn PRUNE(R) :
  Initialize *rules* $= \lambda C.\emptyset$, *proved* $= \emptyset$, *assumed* $= \emptyset$, $\mathcal{I} = \emptyset$
  for U if $R(U) \neq \emptyset$:
    for $C_1, C_2 \in R(U), C_1 \neq C_2$:
      Let $[\![\sigma^c]\!] C_1 [\![\sigma^c]\!]^\dagger = \textsc{CanonC}(C_1)$ // CanonC (4.1)
      $\mathit{rules}(\textsc{CanonC}(C_1)) \leftarrow \mathit{rules}(\textsc{CanonC}(C_1)) \cup \{[\![\sigma^c]\!] C_2 [\![\sigma^c]\!]^\dagger\}$
      $\mathcal{I} = \mathcal{I} \cup \{\textsc{CanonI}(C_1 \approx C_2)\}$ // CanonI (4.2)
  for $I \in \mathcal{I}$ increasing by $|I|$:
    invariant *assumed* $\vdash$ *proved* // C.f. Theorem 7
    proved $=$ *proved* $\cup$ TASOSEARCH(I)
  fn TASOSEARCH(I) :
    $Q =$ PriorityQueue(I)
    visited $= \emptyset$
    while $|Q| > 0 \wedge |\mathit{visited}| >=$ searchlimit :
      $I_0 = Q$.pop()
      foreach $\textsc{CanonI}(C_1 \approx C_2) = I_0$:
        for $C_3 \in \mathit{rules}(\textsc{CanonC}(C_1))$ and $\textsc{CanonI}(C_3 \approx C_1) \in \mathit{proved}$:
          $I_1 = \textsc{CanonI}(C_3 \approx C_2)$
          if $\neg$*visited*.insert(I_1) : **continue**
          else if $I_1 \in \mathit{proved}$: **return** *visited*
          else if $|I_1| < |I|$: **return** *visited* $\cup$ TASOSEARCH(I_1)
          else: Q.push(I_1)
    assumed $\leftarrow$ *assumed* $\cup \{I\}$
    return *visited*
  return *assumed*

In Algorithm 3, the outer function PRUNE first constructs the set of circuit identities $\mathcal{I}$ and the mapping *rules* from $\mathcal{R}_n^{\leq k}[\mathcal{G}]$. The mapping *rules* maps each canonical circuit $\textsc{CanonC}(C)$ to a set of circuits that are equivalent to C given by $\mathcal{R}_n^{\leq k}[\mathcal{G}]$. Then, our algorithm iterates every identity in set $\mathcal{I}$ in increasing order and tries to prove it using the inner function TASOSEARCH. The function TASOSEARCH performs a best-first search on the space of circuit identities, starting from the input identity I. In each iteration, it pops a minimum identity I_0 from the priority queue Q and tries to find a proved rewrite rule from *rules* that can be applied to I_0 to generate a new identity I_1. If I_1 has already been proved, then the function returns successfully. If I_1 is shorter than I, then

Table 1: Number of Rules

k	2	3	4	5	6	7	8	9
L	6	8	8	8	8	8	8	8
C	11	16	17	17	17	17	17	
C+T	10	13	14	16	17	17	17	
C^*+T	30	28	28	28				
C+$\sqrt{\text{T}}$	15	21	22	22	22	22		
C+$\text{R}_\text{Z}(\pi/3)$	15	21	22	22	22	22		

the function recursively calls itself on I_1 to skip all identities larger than I. Otherwise, it pushes I_1 back into the priority queue for further exploration. If no proof is found within the search limit, the identity I is included in *assumed*.

TasoSearch returns the set of all rules explored during the search, which is then added to the set of proved identities *proved* in the outer function Prune. Finally, after all identities in $\mathcal{I}$ have been processed, the algorithm returns the set of assumed identities *assumed* as the pruned equational theory.

Theorem 7. Algorithm 3 correctly computes an equational theory that can derive all identities in $\text{Id}_n^{\leq 2k}(\mathcal{G})$.

5 Experimental Result

We implemented our synthesis and minimization algorithms in Rust[1] and evaluated them on a machine with 20-core Intel(R) Xeon(R) E5 CPU and 128 GB RAM. We set the floating-point precision to 64 bits and the search limit in Algorithm 3 to 50000. We evaluated our algorithms on six different gate sets, defined as follows:

$$\text{L} = \{X, CX\} \qquad C^*\text{+T} = \{X, Y, Z, CX, CY, CZ, H, T, T^\dagger, S, S^\dagger\}$$

$$C = \{X, CX, H, S, S^\dagger\} \qquad C\text{+}\sqrt{\text{T}} = \{X, CX, H, S, S^\dagger, \sqrt{T}, \sqrt{T^\dagger}\}$$

$$C\text{+T} = \{X, CX, H, T, T^\dagger\} \qquad C\text{+R}_\text{Z}(\pi/3) = \left\{X, CX, H, S, S^\dagger, R_Z\left(\frac{\pi}{3}\right), R_Z\left(-\frac{\pi}{3}\right)\right\}$$

5.1 Synthesis and Pruning Results

In this section, we present our primary experimental results. We ran our synthesis and pruning algorithms on 5 qubits for all six gate sets. Due to computational limitations, the maximum circuit size k explored varies for each gate set, ranging from $k = 5$ for C^*+T to $k = 9$ for L. For each gate set and circuit size, we recorded the number of rewrite rules, synthesized ECCs, and canonical identities. The results appear in the tables below.

Table 1 presents the core result of our work: the size of the pruned equational theory for each gate set. A key observation is the rapid convergence to a small, constant

[1] Will release the code upon publication

Table 3: Synthesized Canonical Identities

k	2	3	4	5	6	7	8	9
L	6	19	140	705	3595	16055	46654	66904
C	12	42	306	2221	18259	168611	1652070	
C+T	11	36	243	1775	14349	129274	1225169	
C^*+T	64	645	9583	172251				
C+$\sqrt{\text{T}}$	18	72	496	3989	36330	366327		
C+$R_Z(\pi/3)$	18	77	499	4085	37040	372194		

number of rules. For instance, the L gate set stabilizes at 8 rules for $k \geq 3$, and the C+T gate set stabilizes at 17 rules for $k \geq 6$; these 17 rules are shown in Figure 1. This convergence across different gate sets strongly supports our central hypothesis that a small, finite equational theory is sufficient for describing the vast majority of circuit identities up to a practical size. The larger number of rules for gate sets like C^*+T (30 rules at $k = 5$) is expected due to the richer set of available gates.

Table 3 lists the total number of canonical circuit identities before pruning (in $\mathcal{I}$ of Algorithm 3). When compared with Table 1, the data reveals the power of our pruning algorithm. For example, at $k = 8$ for the C+T gate set, our method discovered 848,880 canonical identities, which were then proven to be derivable from a core set of just 17 rules. This massive reduction in size demonstrates that the vast majority of identities found through brute-force enumeration are redundant. Our approach effectively distills this redundancy into a compact and powerful equational theory.

5.2 Correctness

Ensuring the correctness of the synthesis and pruning process is paramount, especially given its reliance on floating-point arithmetic to represent unitary matrices. Therefore, we conducted several validation experiments to verify the accuracy and reliability of our implementation. These experiments include a comparison with a naive implementation, proving existing quantum optimizers, assessing the impact of floating-point precision, and generating exportable rules and proofs.

Comparison with a Naive Implementation We first compared our optimized implementation with a naive baseline. The baseline uses standard 64-bit floating-point numbers (`f64`) and a fixed-epsilon equality test for unitaries, and it does not use the equivalence classes from Section 3. Tables 7, and 5 report the results. Because the naive method is slow and memory-intensive, we could only run it on smaller circuit sizes; entries that are skipped due to our resource limits are marked with "-".

For each synthesized ECCs of the naive method, we checked if it exists in the results of our optimized implementation. We found that all unitaries and circuits synthesized by the naive method has a corresponding entry in our optimized results for all gate sets and circuit sizes where the naive method completed. This cross-validation confirms our optimized synthesis algorithm matches the naive approach.

Table 5 shows the number of synthesized canonical identities from the naive method. We observe it is a lot larger than those from our optimized implementation (Table 1).

Table 5: Number of Naive-Synthesized Canonical Identities

k	2	3	4	5	6	7
L	5	16	202	2958	37563	438482
C	11	36	372	5449	79392	-
C+T	10	30	311	4758	71001	-
C^*+T	73	821	21493	-		
C+$\sqrt{\text{T}}$	17	63	558	8000	-	-
C+$\text{R}_\text{Z}(\pi/3)$	17	68	563	8088	-	-

Table 7: Number of Irreducible Rules after Adding Naive Rules

k	2	3	4	5	6	7
L	6	8	8	8	8	8
C	12	16	17	17	17	-
C+T	10	13	14	16	17	-
C^*+T	30	28	28	-		
C+$\sqrt{\text{T}}$	16	21	22	22	-	-
C+$\text{R}_\text{Z}(\pi/3)$	16	21	22	22	-	-

Table 9: Proving Rules from Quartz [23]

Quartz($q = 3$)	L	C	C+T	C^*+T	C+$\sqrt{\text{T}}$	C+$\text{R}_\text{Z}(\pi/3)$[2]
k	7	7	7	6	6	6
ECCs	340	103135	65106	610684	27437	580196
Identities	525	116040	70876	785177	30059	712823
Synth Time (s)	0.34	298.81	367.84	1101.27s	162.91	9974.40
Prove Time (s)	17.91	3466.2	1950.6	30917	519.67	11638.2

However, the rules synthesized by the naive method are all derivable from those synthesized by our optimized implementation for all gate sets and circuit sizes where the naive method completed. This validates the correctness of our pruning algorithm.

After we obtain our synthesized and pruned result, we continue to add naive-synthesized identities to see if any of them can not be pruned. Table 7 shows we found that for $k \geq 3$, no additional irreducible rules from the naive method can be added into our optimized pruned set. The additional irreducible rules for $k = 2$ are due to the limited pruning power caused by the rewrite rule set, *rules*, being too small in Algorithm 3. Specifically, if rewrite rules from 3 gates to 1 gates are not added into *rules*, many optimizations cannot be applied in our pruning algorithm.

Proving Existing Quantum Optimizations Second, we validated our synthesized equational theories by using them to prove the rewrite rules employed in an existing quantum circuit optimizer, Quartz [23]. By demonstrating that our synthesized rules can derive those used in Quartz, we provide strong evidence for the practical utility of our approach. We extracted the rewrite rules used in Quartz for various gate sets and attempted to prove each rule using our synthesized equational theories.

Table 9 summarizes the results of this validation. We ran Quartz on 3 qubits and at most 7 gates for most gate sets, as it is the setting used in the original Quartz paper; however, the last 3 gate sets give out-of-memory for this setting, so we $k = 6$ for them. We then attempted to prove each of the rewrite rules used in Quartz using our synthesized equational theories. We proved all the rules used in Quartz for all gate sets, which showcases the level of completeness of our synthesized equational theories. The synthesis and proving times are also reported in Table 9, showing that our methods are efficient enough to handle practical sets of rewrite rules.

[2] Due to bugs in the GitHub version of Quartz for this gate set, we use the artifact version without pruning, which produces many more ECCs compared to C+$\sqrt{\text{T}}$.

Table 11: Number of ECCs for C+T($k = 7$) under `f64`/`f128` and different comparison precisions (p).

p	8	16	24	32	40	48
`f64`	5658695	5658695	5658695	5658695	5660176	6431388
`f128`	5658695	5658695	5658695	5658695	5658695	5658695

Impact of Floating-Point Precision Third, we investigate the impact of floating-point precision on the synthesis results. In our main experiments, we used 64-bit floating-point numbers to represent unitary matrices, which may raise concerns about numerical stability and correctness. To address this, we ran experiments on the C+T gate set with $k = 7$ using both standard 64-bit (`f64`) and 128-bit (`f128`) floating-point numbers. In these experiments, we varied the comparison precision threshold, p, which dictates how many of the most significant bits of the floating-point representation of the state vectors must match to be considered equal in Algorithm 2. This parameter is crucial for balancing comparison precision against numerical noise sensitivity.

The results are shown in Table 11. We observe that for comparison precisions p up to 32 bits, both `f64` and `f128` yield the same number of synthesized ECCs (5658695). This consistency indicates that our synthesis results are robust to floating-point precision within this range. However, as we increase p beyond 32 bits, discrepancies begin to emerge. At $p = 40$, `f64` produces a slightly different count (5660176) compared to `f128` (5658695), and at $p = 48$, the difference becomes more pronounced (6,431,388 for `f64` vs. 5,658,695 for `f128`). These are caused by numerical inaccuracies in `f64` arithmetic that lead to incorrect equivalence classifications.

Exportable Rules and Proofs A key benefit of our approach is that the final, pruned set of rules is small enough for manual inspection. For example, the 17 rules generated for the C+T gate set (shown in Figure 1) are simple and can be individually verified. Since our framework proves that all other synthesized identities can be derived from this core set, verifying these base rules provides strong evidence for the correctness of the thousands of identities they represent.

To provide stronger confidence, and to inspect the correctness of our pruning algorithm (Algorithm 3) itself, our implementation can generate a formal, human-readable proof for any pruned identity. This feature makes the entire process transparent and allows for independent verification, ensuring that our results are both trustworthy and understandable. An example of such a proof is provided in Appendix B.

5.3 Synthesis and Pruning Time

In addition to the synthesis and pruning results presented earlier, we also measured the time taken for both processes across different gate sets and circuit sizes, focusing on the C+T gate set as a representative example. Table 13 summarizes the time taken (in seconds) for the pruning and synthesis stages of our algorithm, as well as the corresponding times for a naive implementation that does not utilize equivalence classes or optimized floating-point comparisons.

From the data, we observe that our optimized implementation significantly outperforms the naive approach, especially as the circuit size k increases. For instance,

Table 13: Time for Each Stages of our Algorithm for gate set C+T. (s)

k	2	3	4	5	6	7	8
Synth	0.01	0.03	0.17	1.17	10.76	109.13	1119.40
Prune	0.03	0.83	1.47	12.59	258.15	7274.39	149602.22
Naive Synth	0.00	0.06	1.32	23.21	391.60	-	
Check Equiv	0.18	2.35	42.12	772.77	1145.84	-	
Naive Prune	0.01	0.11	1.42	42.05	2483.19	-	
Quartz Synth	0.08	0.84	15.68	327.43	-		

at $k = 6$, our pruning process takes approximately 258 seconds, while the naive method takes over 2375 seconds. The synthesis times also show a similar trend, with our method being substantially faster. Our method is also more memory efficient, allowing us to complete synthesis for $k = 7$ and $k = 8$, where the naive approach runs out of memory. We also report the synthesis time of Quartz [23] in the same settings (5 qubits). Quartz runs out of memory for $k \geq 6$ and is much slower than the naive baseline, likely due to general matrix multiplication rather than our SV-Sim-style gate-by-gate approach [17].

6 Related Work

Recent years have seen numerous quantum circuit optimizers, including heuristic methods [20,15,5] and formal methods [23,11,13,12,4]. Our work is most closely related to formal methods that leverage rewrite rules, such as Quartz [23] and QUESO [22]. These approaches typically target circuit-level optimization by synthesizing rewrite rules or equivalent circuits, without understanding and reasoning about the the underlying equational theory. In contrast, we focus on synthesizing a minimal set of identities that can derive all equalities up to a given size, reducing the complexity of the underlying equational theory.

Theoretical work has also explored complete equational theories for quantum circuits [2,10,18,19,1,14,9,8,6,7], establishing minimal and complete generators for various gate sets. However, algebraic completeness does not guarantee practical utility for optimization or synthesis. Our work bridges this gap by identifying which rules are essential and sufficient in realistic settings, showing that a small, tractable set of identities suffices to capture the circuit behavior relevant for practical applications.

7 Conclusion

We presented algorithms for synthesizing and pruning equational theories of quantum circuits. Our experiments show that, for several finite gate sets, a small set of identities suffices to derive all equalities up to practically relevant sizes, revealing a surprisingly compact structure underlying circuit identities. These findings not only shed light on practical quantum compiling but also open the door to a deeper exploration of the algebraic structure of quantum circuit equalities. Future work includes (i) applying our methods to practical quantum compilers to evaluate their optimization impact, and (ii) investigating the deeper structure of equalities, including questions of finiteness and redundancy.

References

1. Amy, M., Chen, J., Ross, N.J.: A finite presentation of CNOT-dihedral operators. In: Coecke, B., Kissinger, A. (eds.) Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL 2017, Nijmegen, The Netherlands, 3–7 July 2017. Electronic Proceedings in Theoretical Computer Science, vol. 266, pp. 84–97 (2018). https://doi.org/10.4204/EPTCS.266.5, arXiv:1701.00140
2. Bian, X., Selinger, P.: Generators and relations for 2-qubit Clifford+T operators. In: Proceedings of the 18th International Conference on Quantum Physics and Logic, QPL 2022. Electronic Proceedings in Theoretical Computer Science, vol. 394, pp. 13–28 (2023). https://doi.org/10.4204/EPTCS.394.2, arXiv:2204.02217
3. Bravyi, S., Latone, J.A., Maslov, D.: 6-qubit optimal clifford circuits. npj Quantum Information **8**(1), 79 (2022). https://doi.org/10.1038/s41534-022-00583-7, https://doi.org/10.1038/s41534-022-00583-7
4. Chareton, C., Bardin, S., Bobot, F., Perrelle, V., Valiron, B.: An automated deductive verification framework for circuit-building quantum programs. In: Programming Languages and Systems (ESOP 2021). Lecture Notes in Computer Science, vol. 12648, pp. 148–177. Springer (2021). https://doi.org/10.1007/978-3-030-72019-3_6, arXiv:2003.05841
5. Cirq Developers: Cirq (2024). https://doi.org/10.5281/zenodo.4062499, https://github.com/quantumlib/Cirq, see https://quantumai.google/cirq
6. Clément, A., Delorme, N., Perdrix, S.: Minimal equational theories for quantum circuits. In: Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3661814.3662088, https://doi.org/10.1145/3661814.3662088
7. Clément, A., Heurtel, N., Mansfield, S., Perdrix, S., Valiron, B.: A complete equational theory for quantum circuits. In: 2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–13 (2023). https://doi.org/10.1109/LICS56636.2023.10175801
8. Cockett, J.R.B., Comfort, C.: The category TOF. In: Selinger, P., Chiribella, G. (eds.) Proceedings of the 15th International Conference on Quantum Physics and Logic, QPL 2018, Halifax, Canada, 3–7 June 2018. Electronic Proceedings in Theoretical Computer Science, vol. 287, pp. 67–84 (2019). https://doi.org/10.4204/EPTCS.287.4, arXiv:1804.10360
9. Cockett, J.R.B., Comfort, C., Srinivasan, P.V.: The category CNOT. In: Coecke, B., Kissinger, A. (eds.) Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL 2017, Nijmegen, The Netherlands, 3–7 July 2017. Electronic Proceedings in Theoretical Computer Science, vol. 266, pp. 258–293 (2018). https://doi.org/10.4204/EPTCS.266.18, arXiv:1707.02348
10. Coecke, B., Wang, Q.: ZX-rules for 2-qubit Clifford+T quantum circuits. In: Reversible Computation – 10th International Conference, RC 2018, Leicester, UK, September 12–14, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11106, pp. 144–161. Springer (2018). https://doi.org/10.1007/978-3-319-99498-7_10, arXiv:1804.05356
11. Cowtan, A., Dilkes, S., Duncan, R., Krajenbrink, A., Simmons, W., Sivarajah, S.: On the qubit routing problem. In: 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 135, pp. 5:1–5:32 (2019). https://doi.org/10.4230/LIPIcs.TQC.2019.5, arXiv:1902.08091
12. Davis, M.G., Smith, E., Tudor, A., Sen, K., Siddiqi, I., Iancu, C.: Towards optimal topology aware quantum circuit synthesis. In: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). pp. 223–234. IEEE (2020). https://doi.org/10.1109/QCE49297.2020.00036, best Paper Award at IEEE Quantum Week 2020

13. Hietala, K., Rand, R., Hung, S.H., Wu, X., Hicks, M.: A verified optimizer for quantum circuits. Proceedings of the ACM on Programming Languages **5**(POPL), 1–29 (2021). https://doi.org/10.1145/3434318, arXiv:1912.02250, Distinguished Paper at POPL 2021
14. Iwama, K., Kambayashi, Y., Yamashita, S.: Transformation rules for designing CNOT-based quantum circuits. In: Proceedings of the 39th Annual Design Automation Conference, DAC 2002, New Orleans, LA, USA, June 10–14, 2002. pp. 419–424. ACM (2002). https://doi.org/10.1145/513918.514026
15. Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C.J., Lishman, J., Gacon, J., Martiel, S., Nation, P.D., Bishop, L.S., Cross, A.W., Johnson, B.R., Gambetta, J.M.: Quantum computing with Qiskit. arXiv preprint arXiv:2405.08810 (2024), arXiv:2405.08810
16. Jia, Z., Padon, O., Thomas, J., Warszawski, T., Zaharia, M., Aiken, A.: Taso: Optimizing deep learning computation with automatic generation of graph substitutions. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles. pp. 47–62. SOSP '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341301.3359630, https://doi-org.ezproxy.lib.purdue.edu/10.1145/3341301.3359630
17. Li, A., Krishnamoorthy, S.: Sv-sim: Scalable pgas-based state vector simulation of quantum circuits. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (2021)
18. Makary, J., Ross, N.J., Selinger, P.: Generators and relations for real stabilizer operators. In: Heunen, C., Backens, M. (eds.) Proceedings of the 18th International Conference on Quantum Physics and Logic, QPL 2021, Gdansk, Poland, and online, 7–11 June 2021. Electronic Proceedings in Theoretical Computer Science, vol. 343, pp. 14–36 (2021). https://doi.org/10.4204/EPTCS.343.2, arXiv:2109.05655
19. Ranchin, A., Coecke, B.: Complete set of circuit equations for stabilizer quantum mechanics. Physical Review A **90**(1), 012109 (2014). https://doi.org/10.1103/PhysRevA.90.012109, arXiv:1310.7932
20. Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., Duncan, R.: t|ket⟩: A retargetable compiler for NISQ devices. Quantum Science and Technology **5**(1), 014003 (2020). https://doi.org/10.1088/2058-9565/ab8e92, arXiv:2003.10611
21. Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., Duncan, R.: t|ket⟩: A retargetable compiler for NISQ devices. Quantum Science and Technology **5**(1), 014003 (2020). https://doi.org/10.1088/2058-9565/ab8e92, arXiv:2003.10611
22. Xu, A., Molavi, A., Pick, L., Tannu, S., Albarghouthi, A.: Synthesizing quantum-circuit optimizers. vol. 7. Association for Computing Machinery, New York, NY, USA (Jun 2023). https://doi.org/10.1145/3591254, https://doi.org/10.1145/3591254
23. Xu, M., Li, Z., Padon, O., Lin, S., Pointing, J., Hirth, A., Ma, H., Palsberg, J., Aiken, A., Acar, U.A., Jia, Z.: Quartz: Superoptimization of quantum circuits. In: Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation. p. 625–640. PLDI 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3519939.3523433, https://doi.org/10.1145/3519939.3523433

A Missing Proofs

Theorem 1 (QCEL Soundness). For any equational theory Γ over a quantum gate set $\mathcal{G}$ and any n-qubit quantum circuit C, if $\Gamma \vdash C \approx id_n$ and $\models \Gamma$, then $\models C \approx id_n$.

Proof (Proof of Theorem 1). The proof proceeds by induction on the derivation of $\Gamma \vdash C \approx id_n$. We show that each inference rule in Figure 2 preserves validity.

- Eq: Trivial because $[\![id_n]\!] = I^{\otimes n}$.
- Assume: If $C \approx id_n \in \Gamma$, since $\models \Gamma$, we have $\models C \approx id_n$ by definition.
- Adjoint: Assume $\models C \approx id_n$, i.e., $[\![C]\!] = e^{i\alpha} I$. Then $[\![C^\dagger]\!] = [\![C]\!]^\dagger = (e^{i\alpha} I)^\dagger = e^{-i\alpha} I$, so $\models C^\dagger \approx id_n$.
- Extend: Assume $\models C \otimes id_m \approx id_{n+m}$. Then $[\![C]\!] \otimes I^{\otimes m} = e^{i\alpha} I^{\otimes (n+m)} = (e^{i\alpha} I^{\otimes n}) \otimes I^{\otimes m}$. This implies $[\![C]\!] = e^{i\alpha} I^{\otimes n}$. Consequently, $[\![C \otimes id_k]\!] = [\![C]\!] \otimes I^{\otimes k} = e^{i\alpha} I^{\otimes (n+k)}$, so the conclusion holds.
- Rotate: Assume $\models C_1 \circ C_2 \approx id_n$, so $[\![C_1]\!][\![C_2]\!] = e^{i\alpha} I$. Multiplying by $[\![C_1]\!]^\dagger$ on the left gives $[\![C_2]\!] = e^{i\alpha} [\![C_1]\!]^\dagger$. Then $[\![C_2]\!][\![C_1]\!] = e^{i\alpha} [\![C_1]\!]^\dagger [\![C_1]\!] = e^{i\alpha} I$, so $\models C_2 \circ C_1 \approx id_n$.
- Permutation: Assume $\models C \approx id_n$. Then $[\![\sigma^c \circ C \circ (\sigma^c)^\dagger]\!] = [\![\sigma^c]\!][\![C]\!][\![\sigma^c]\!]^\dagger = [\![\sigma^c]\!](e^{i\alpha} I)[\![\sigma^c]\!]^\dagger = e^{i\alpha} [\![\sigma^c]\!][\![\sigma^c]\!]^\dagger = e^{i\alpha} I$.
- Transitivity: Assume $\models C_1 \circ C_2^\dagger \approx id_n$ and $\models C_2 \circ C_3^\dagger \approx id_n$. Then $[\![C_1]\!][\![C_2]\!]^\dagger = e^{i\alpha} I$ and $[\![C_2]\!][\![C_3]\!]^\dagger = e^{i\beta} I$. Thus $[\![C_1]\!] = e^{i\alpha} [\![C_2]\!]$ and $[\![C_2]\!] = e^{i\beta} [\![C_3]\!]$. Substituting gives $[\![C_1]\!] = e^{i(\alpha+\beta)} [\![C_3]\!]$, or $[\![C_1]\!][\![C_3]\!]^\dagger = e^{i(\alpha+\beta)} I$, which means $\models C_1 \approx C_3$.

□

Proof (Proof of Lemma 2). ($\supseteq$) Trivial based on definition. ($\subseteq$) Consider any identity $C' \approx id_n$ in $\mathrm{Id}_n^{2k}(\mathcal{G})$. By definition, $[\![C']\!] = id_n$ and the size of C' is at most $2k$. We can decompose C' into two circuits C_1' and C_2' such that $C' = C_1' \circ C_2'^\dagger$ and both C_1' and C_2' have size at most k. Let $U = \textsc{Reduce}([\![C_1']\!]) = \textsc{Reduce}([\![C_2']\!])$. By Lemma 1, there exist circuits $C_1 \in \mathcal{R}_n^{\leq k}[\mathcal{G}](U)$ and symmetries $\sigma_1 \sigma_2 \in \mathrm{Sym}(n)$ such that $C_1' = \sigma_1^c \circ C_1 \circ \sigma_2^c$. One can also check that C_2 defined by $C_2' = \sigma_1^c \circ C_2 \circ \sigma_2^c$, also satisfying $C_2 \in \mathcal{R}_n^{\leq k}[\mathcal{G}](U)$ since $[\![C_1']\!] = [\![C_2']\!]$. Therefore, using C_1 and C_2, we have $C' = \sigma_1^c \circ C_1 \circ \sigma_2^c \circ (\sigma_2^c)^\dagger \circ C_2^\dagger \circ (\sigma_1^c)^\dagger = \sigma_1^c \circ C_1 \circ C_2^\dagger \circ (\sigma_1^{-1})^c$ using (2.2). Thus, $C' \approx id_n$ can be expressed in the desired form. □

Theorem 2. Algorithm 1 correctly computes the mapping $\mathcal{R}_n^{\leq k}[\mathcal{G}]$ for the given quantum gate set $\mathcal{G}$, circuit size k, and number of qubits n.

Proof (Proof of Theorem 2). We prove the correctness by induction on the circuit size i. For the base case $i = 1$, the algorithm initializes R^1 to contain all single-gate circuits from $\mathcal{G}$, which is correct by definition. For the inductive step, assume that R^{i-1} correctly computes $\mathcal{R}_n^{i-1}[\mathcal{G}]$. We need to show that R^i correctly computes $\mathcal{R}_n^i[\mathcal{G}]$.

($\subseteq$) Consider any unitary $U' \in \mathrm{U}(2^n)$ such that $R^i(U') \neq \emptyset$. By the construction of R^i, there exists a gate $G \in \mathrm{Circ}_n^1(\mathcal{G})$ and a unitary $U \in \mathrm{U}(2^n)$ such that $U' = \textsc{Reduce}([\![G]\!]\, U)$ and $R^{i-1}(U) \neq \emptyset$. By the inductive hypothesis, $R^{i-1}(U)$ contains all circuits of size $i-1$ that evaluate to the canonical representative of U. Therefore, for

each circuit $C \in R^{i-1}(U)$, the circuit $\sigma_2^{\mathrm{c}} \circ G \circ C \circ \sigma_1^{\mathrm{c}}$ evaluates to U' and has size i. Thus, all circuits in $R^i(U')$ are of size i and evaluate to the canonical representative of U'.

($\supseteq$) Consider any circuit $C' \in \mathrm{Circ}_n^i(\mathcal{G})$ such that $[\![C']\!] = \textsc{Reduce}([\![C']\!]) = U'$. By the definition of circuit construction, there exists a gate $G' \in \mathrm{Gen}_n(\mathcal{G})$ and a circuit C'' such that $C' = G' \circ C''$. Since $C'' \in \mathrm{Circ}_n^{i-1}(\mathcal{G})$, by Lemma 1, there exists a circuit C and symmetries σ_1, σ_2 such that $C'' = \sigma_2^{\mathrm{c}} \circ C \circ \sigma_1^{\mathrm{c}}$ and $C \in \mathcal{R}_n^{i-1}[\mathcal{G}](U)$ where $U = \textsc{Reduce}(C'')$. Let $G = (\sigma_2^{-1})^{\mathrm{c}} \circ G' \circ \sigma_2^{\mathrm{c}}$, by (2.3), $G \in \mathrm{Gen}_n(\mathcal{G})$. Then we have $C' = \sigma_2^{\mathrm{c}} \circ G \circ C \circ \sigma_1^{\mathrm{c}}$. By the inductive hypothesis, $C \in R^{i-1}(U)$. Therefore, during the construction of R^i, the circuit C' will be added to $R^i(U')$ where $U' = \textsc{Reduce}([\![G]\!]\, U)$. Thus, all circuits of size i that evaluate to the canonical representative of U' are included in $R^i(U')$.

By induction, the algorithm correctly computes $\mathcal{R}_n^{\leq k}[\mathcal{G}]$. □

Theorem 3. Given a unitary U on n qubits, and a fully randomized quantum state $|\psi_r\rangle$ independent to U, $\Pr\left(\left|\mathrm{Sym}^{\approx}\left(\text{deφase } U|\psi_r\rangle\right)\right| > 1\right) = 0$.

Proof (Proof of Theorem 3). Given a Haar-random quantum state $|\psi_r\rangle \sim \mathrm{Haar}(\mathbb{C}_2^{\otimes n})$. Let $|\psi\rangle = U|\psi_r\rangle$, obviously, $|\psi\rangle$ is also Haar-random. Since $\left|\mathrm{Sym}^{\approx}\left(\text{deφase } U|\psi_r\rangle\right)\right| > 1$ implies that there exist two different qubits $1 \leq i < j \leq n$ such that one of $\langle\psi|2^i\rangle = \langle\psi|2^j\rangle$, and $\langle\psi|2^n - 1 - 2^i\rangle = \langle\psi|2^n - 1 - 2^j\rangle$ holds. Note that the deφase operation does not affect these equalities. Since $|\psi\rangle$ is Haar-random, the probability of this event happening is zero. Therefore, we have $\Pr\left(\left|\mathrm{Sym}^{\approx}\left(\text{deφase } U|\psi_r\rangle\right)\right| > 1\right) = 0$. □

Theorem 4. Given two unitaries U_1 and U_2, and a fully randomized quantum state $|\psi_r\rangle$ independent to both U_1 and U_2, if deφase $U_1|\psi_r\rangle$ = deφase $U_2|\psi_r\rangle$ holds, then the posterior probability $\Pr\left(\exists\alpha.\ U_1 = e^{\mathrm{i}\alpha}U_2\right) = 1$.

Proof (Proof of Theorem 4). Let events $A = \{\text{deφase } U_1|\psi_r\rangle = \text{deφase } U_2|\psi_r\rangle\}, B = \{\exists\alpha.\ U_1 = e^{\mathrm{i}\alpha}U_2\}$, and $V_1, \cdots, V_m$ are eigenspaces of $U_2^\dagger U_1$. We have,

$$\begin{aligned}\Pr(A|\neg B) &= \Pr\left(\exists\alpha.\ U_1|\psi_r\rangle = e^{\mathrm{i}\alpha}U_2|\psi_r\rangle \middle| \neg B\right) = \Pr\left(\exists\alpha.\ U_2^\dagger U_1|\psi_r\rangle = e^{\mathrm{i}\alpha}|\psi_r\rangle \middle| \neg B\right) \\ &= \sum_{k=1}^{m} \Pr(|\psi_r\rangle \in V_k|\neg B) = 0\end{aligned}$$

The last equality holds because if $\neg B$ holds, then $U_2^\dagger U_1 - e^{\mathrm{i}\alpha}$ is not a zero operator for any α, thus each eigenspace V_k has a dimension less than 2^n. Thus, by Bayes' theorem, we have $\Pr(B|A) = 1$. □

Theorem 5. Algorithm 2 returns $U_{\min}$ satisfying (3.1) under the assertion and assumption.

Proof (Proof of Theorem 5). ($\Rightarrow$) Trivial due to line 9. ($\Leftarrow$) For any $\sigma_1, \sigma_2 \in \mathrm{Sym}(n)$, let $U' = [\![\sigma_2^{\mathrm{c}}]\!] U [\![\sigma_1^{\mathrm{c}}]\!]$, we compare each line of Algorithm 2 to show that $\textsc{Reduce}(U') = \textsc{Reduce}(U)$. Variables in the algorithm of $\textsc{Reduce}(U')$ are denoted with $\cdot'$, to distinguish from those in $\textsc{Reduce}(U)$.

- At line 3, we can see $|\psi_0'\rangle = \text{deφase } [\![\sigma_1^{\mathrm{c}}]\!]^\dagger U^\dagger [\![\sigma_2^{\mathrm{c}}]\!]^\dagger |\psi_l\rangle = [\![\sigma_1^{\mathrm{c}}]\!]^\dagger \text{deφase } U^\dagger|\psi_l\rangle = [\![\sigma_1^{\mathrm{c}}]\!]^\dagger|\psi_0\rangle$, based on the definition of deφase and ψ_l.

- At line 4, from the definition of $\mathrm{Sym}^{\approx}$, we can see that $\mathrm{Sym}^{\approx}(\llbracket\sigma_1^c\rrbracket^\dagger\psi_0) = \mathrm{Sym}^{\approx}(\psi_0)\cdot\sigma_1^c$. Therefore, for each $\sigma_r' \in \mathrm{Sym}^{\approx}(\psi_0')$, there exists a unique $\sigma_r \in \mathrm{Sym}^{\approx}(\psi_0)$ such that $\sigma_r' = \sigma_r \cdot \sigma_1$, then we compare every two corresponding iterations based on σ_r and σ_r'.
- At line 5, we have $|\psi'\rangle = \text{deφase}\ \llbracket\sigma_2^c\rrbracket U\llbracket\sigma_1^c\rrbracket\llbracket\sigma_1^c\rrbracket^\dagger\llbracket\sigma_r^c\rrbracket^\dagger|\psi_r\rangle = \llbracket\sigma_2^c\rrbracket|\psi\rangle$.
- At line 7, using the definition of $\mathrm{Sym}^{\approx}$ again, we have $\sigma_l' = \sigma_l\sigma_2^{-1}$.
- Finally, the algorithm yields the same set of $\llbracket\sigma_l^c\rrbracket\psi$ and $\llbracket\sigma_l^c\rrbracket U\llbracket\sigma_r^c\rrbracket$. Using the assumption at line 11, the same minimum unitary $U'_{\min} = U_{\min}$ is returned. □

Theorem 6. If the assertion and assumption hold, Algorithm 2 returns

$$\Sigma = \{(\sigma_l, \sigma_r) \mid \exists\alpha,\ \sigma_l U\sigma_r = e^{\mathrm{i}\alpha}U_{\min}\}.$$

Proof (Proof of Theorem 6). (⊆) Trivial due to assumption at line 11. (⊇) For those symmetries $\sigma_r \notin \mathrm{Sym}^{\approx}(\psi_0)$ filtered out at line 4, since deφase $U_{\min}^\dagger|\psi_l\rangle$ is seemingly ordered while deφase $\llbracket\sigma_r^c\rrbracket U^\dagger\llbracket\sigma_l^c\rrbracket^\dagger|\psi_l\rangle$ is not, the corresponding unitary $\llbracket\sigma_l^c\rrbracket U\llbracket\sigma_r^c\rrbracket^\dagger$ are guaranteed to be different from $U_{\min}$ up to a certain phase factor. For any symmetries $\sigma_l \in \mathrm{Sym}(n)$, $\sigma_r \in \mathrm{Sym}^{\approx}(\psi)$ that doesn't satisfy the condition at line 10, because deφase $U_{\min}|\psi_r\rangle \neq$ deφase $\llbracket\sigma_l^c\rrbracket U\llbracket\sigma_r^c\rrbracket^\dagger|\psi_r\rangle$, $U_{\min}$ and $\llbracket\sigma_l^c\rrbracket U\llbracket\sigma_r^c\rrbracket^\dagger$ are also different up to a certain phase factor. □

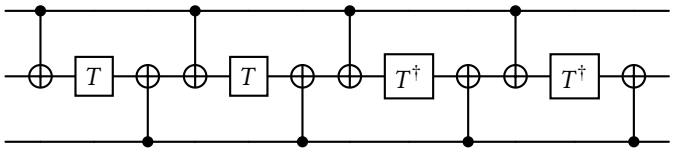
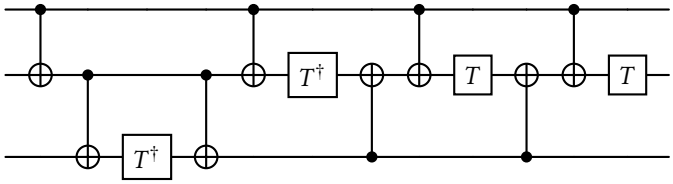
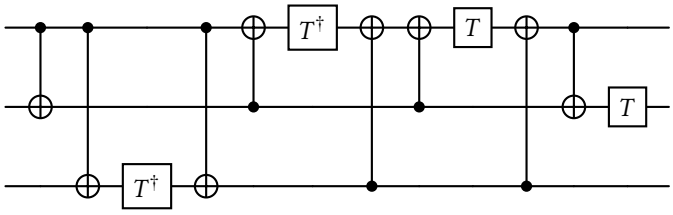
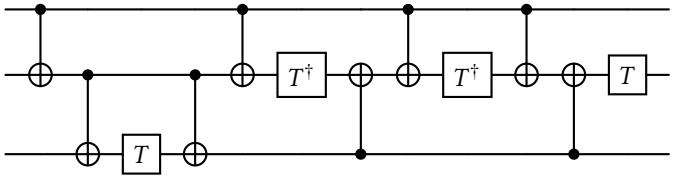
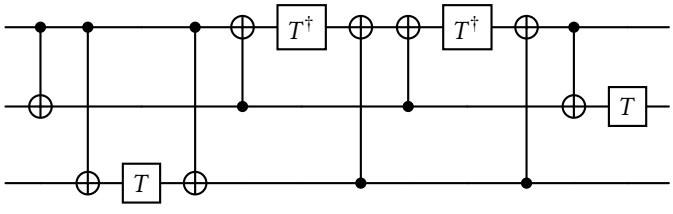
Theorem 7. Algorithm 3 correctly computes an equational theory that can derive all identities in $\mathrm{Id}_n^{\leq 2k}(\mathcal{G})$.

Proof (Proof of Theorem 7). First we prove the invariant at line 9, i.e., all identities in *proved* can be derived from the identities in *assumed*. This is true at the beginning since both sets are empty. Now consider any identity I added to *proved* during the iteration. In TasoSearch, every I_1 added into *visited* set can be derived from and to I using the proved identities in *proved*. Therefore, 1) if one of the I_1 is already in *proved*, the entire *visited* can also be derived from the identities in *assumed*. 2) If the search exceeds the limit, adding I to *assumed* also leads to *assumed* $\vdash$ *visited*. 3) If I_1 is smaller than I, the *visited* set inside TasoSearch(I_1) procedure are also equivalent to proving I. Therefore, *assumed* $\vdash$ *visited* and the invariant holds.

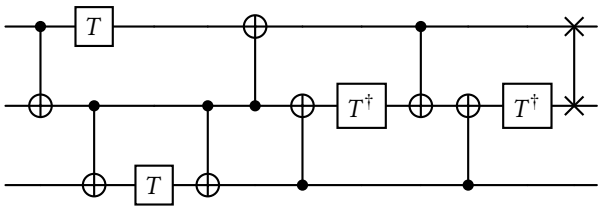
Since $\mathcal{I} \subseteq$ *proved*, and all identities in $\mathrm{Id}_n^{\leq 2k}(\mathcal{G})$ can be derived using identities in $\mathcal{I}$ using Permutation and Rotate rules due to Lemma 2, all identities in $\mathrm{Id}_n^{\leq 2k}(\mathcal{G})$ can also be derived from the identities in *assumed*. □

B Example of an Exported QCEL Proof

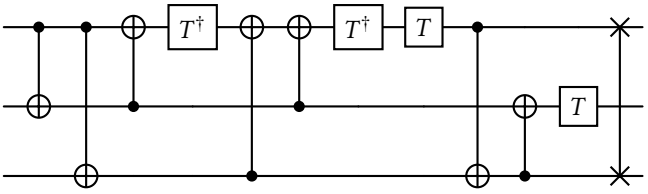
For simplicity, we only show Transitivity rule application in this example. We leave the application of other rules implicit to the readers.

Rule	Canonical Identity
$0 \leftarrow 1, 26$	
$1 \leftarrow 2, 23$	
$2 \leftarrow 3, 25$	
$3 \leftarrow 4, 25$	
$4 \leftarrow 5, 19$	

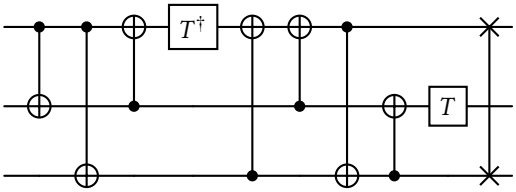
5 ← 6, 11



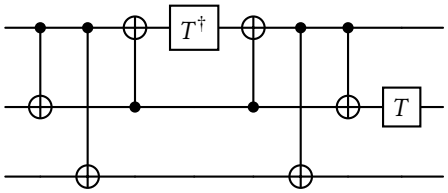
6 ← 7, 18



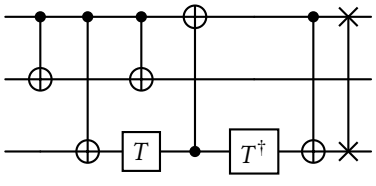
7 ← 8, 15

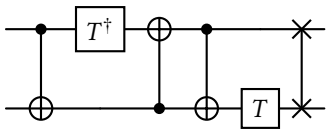
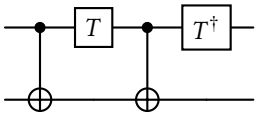
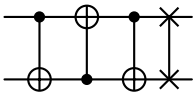
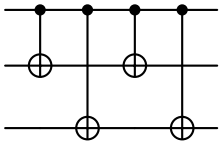
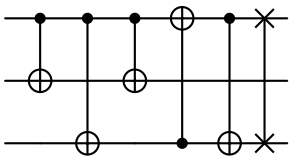
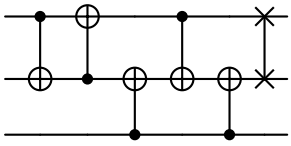


8 ← 9, 14

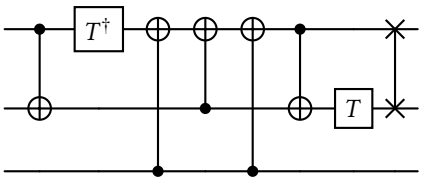


9 ← 10, 13

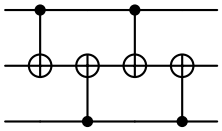


10 ← 11, 12	
11 ← ASSUME	
12 ← ASSUME	
13 ← ASSUME	
14 ← 13, 12	
15 ← 16, 11	

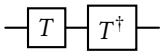
16 ← 10, 17



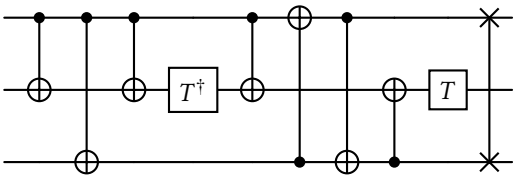
17 ← ASSUME



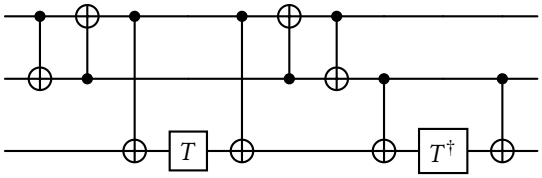
18 ← ASSUME



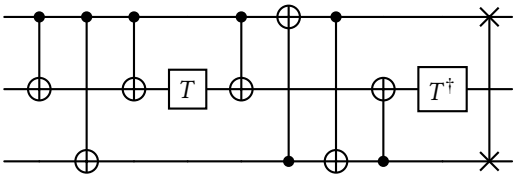
19 ← 20, 12

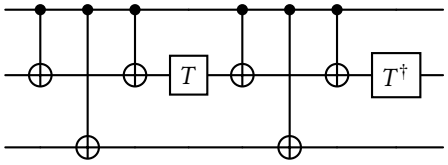
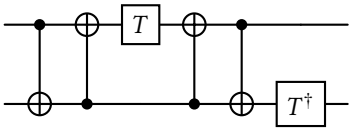
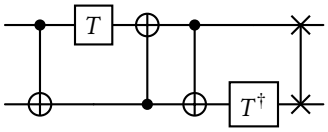
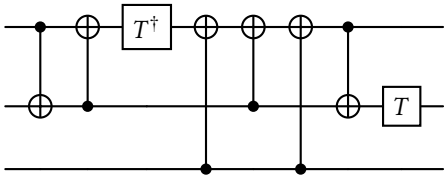
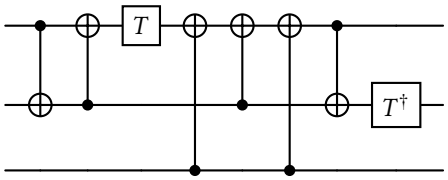


20 ← 21, 12



21 ← 22, 12



$22 \leftarrow 8, 23$	
$23 \leftarrow 24, 12$	
$24 \leftarrow 11, 12$	
$25 \leftarrow 16, 12$	
$26 \leftarrow 27, 12$	

$27 \longleftarrow 24, 17$