Optimizing Quantum Circuits via Letterplace Ideals

Ichio Kikuchi[1], Akihito Kikuchi[2*]

[1]Internationales Forschungszentrum für Quantentechnik

[2]International Research Center for Quantum Technology, Tokyo

March 19, 2026

Abstract

In this article, we present an algebraic approach to optimizing quantum circuits. Our method constructs a Gröbner basis (standard basis) for a set of equations that describe equivalences between gate operations. This allows us to derive additional equivalences up to a given bound and to use the resulting relations as rewriting rules for quantum circuits. When a sufficiently large set of such rules is generated, ambiguities in circuit rewriting—such as obtaining multiple different results—can be avoided.

1 Introduction

Optimizing quantum circuits is a crucial step toward improving the performance of quantum computing systems. It involves rewriting circuit designs by removing redundancies and rearranging sequences of gate operations. In performing this task, the non-commutative nature of quantum gates is the major difficulty: the order of operations is crucial to the semantics of quantum computation, and many gate sequences admit algebraic equivalences that can be exploited for simplification, compression, or structural insight. Traditional approaches to circuit optimization often rely on heuristic rewriting systems or numerical cost functions. In this work, we discuss the feasibility of a symbolic-algebraic approach that treats quantum circuits as elements of a free associative algebra and leverages the theory of non-commutative Gröbner bases to perform systematic simplification. Our approach is grounded in Letterplace correspondence, a powerful method that embeds graded two-sided ideals of the free associative algebra into a commutative polynomial ring with double-indexed variables[6–10, 12, 16, 17]. This embedding enables the computation of non-commutative Gröbner bases using standard commutative algorithms, while preserving the essential non-commutative structure of quantum gate sequences. By encoding quantum gate identities

*akihito_kikuchi@gakushikai.jp (The corresponding author; a visiting researcher in IFQT)

as generators of a two-sided ideal, and applying a symmetry-reduced Buchberger procedure in the letterplace setting, we obtain a canonical form for quantum circuits up to algebraic equivalence.

This paper is constructed as follows. First, it presents the theoretical foundations of this method, illustrates its application to representative quantum circuits, and discusses its potential for integration into quantum compiler tools. Beyond its computational utility, our framework invites a deeper reflection on the algebraic and symbolic nature of quantum computation, where the dance of gates becomes a language of non-commutative geometry.

2 Related Work

2.1 SAT/#SAT depth-optimal synthesis

SAT encodings are used to synthesize depth-optimal Clifford circuits by solving a family of SAT instances parameterized by depth [13]. More recently, quantum circuit synthesis for Clifford+T has been reduced to maximum model counting (#SAT), explicitly supporting depth-optimal exact and approximate synthesis [21].

2.2 ZX-calculus rewriting

ZX-calculus offers a powerful graphical rewriting framework for circuit equivalence and optimization. PyZX implements automated simplification and circuit extraction pipelines [14, 15]. Kissinger and van de Wetering describe T-count reduction via ZX-calculus and phase teleportation, emphasizing nonlocal phase interactions [4, 14].

2.3 Exact synthesis of Clifford+T

Exact synthesis results characterize Clifford+T representable unitaries over rings such as $\mathbb{Z}[1/\sqrt{2}, i]$ and provide constructive algorithms, though not necessarily gate-count optimal [1, 5].

2.4 Template-based simplification

Template-based methods perform local rewriting using precomputed equivalences and are widely applied for reversible quantum circuit simplification [3, 11]. A template is a rule of rewriting (reduction) of the gate sequence, for example.

$$ABC \to D.$$

This rule simplifies a sequence of three gates to a gate. If this pattern is found as a part in a circuit, say,

$$U_1 ABC U_2$$

, the latter is simplified as

$$U_1ABCU_2 \to U_1DU_2.$$

In Qiskit template optimization, templates are given as the sequence of gate operations equivalent to the identity, such as

$$G_1G_2\cdots G_n = 1.$$

By inversion of the head and tail parts of these relations, the following rewriting rules are derived;

$$G_lG_{l+1}\cdots G_{m-1}G_m \to G_{l-1}^{-1}\cdot G_1^{-1}G_n^{-1}G_{n-1}^{-1}\cdot G_{m+1}^{-1}.$$

The algorithm for generating equivalent circuits up to a limited size with a set of specified gates is presented in [20]. These equivalences yield rewriting rules. By carefully selecting rules that reduce circuit size and applying them iteratively, the circuit can be systematically optimized.

A limitation of template-based approaches is the lack of uniqueness in the rewriting results—that is, the absence of the Church–Rosser property, which ensures that different rewriting (or reduction) paths lead to the same final outcome. The approach presented in this article addresses this issue. We show how an algebraic method can generate a set of rewriting rules that allows us to guarantee the Church–Rosser property up to a certain bound.

Moreover, although the algorithm for generating equivalent circuits in [20] is exhaustive, many of the resulting equivalences are not useful for circuit optimization when the target circuit does not contain the corresponding patterns. The computational cost of that algorithm also becomes substantial when equivalences are sought for circuits with many gates and qubits. In contrast, the approach presented in this article begins with a small set of rules that are already effective for optimization and derives more complex rules from them. It is, therefore, useful in intensively seeking the possibility of reduction by a limited number of rewriting rules. For example, let us consider the case with two rules:

$$AB \to D$$

and

$$BC \to E$$

If we have no other rules than these two, what should we do when we find a pattern like

$$ABC \to ?$$

The algebraic approach introduced below helps address this issue.

3 Overview of concepts in the algebraic approach

3.1 Rewriting and uniqueness

Our approach is to assign symbolic representations to quantum circuits (or quantum operators) and to rewrite them so that they attain forms suitable for efficient computation.

First, let us investigate a simple case. Let us rewrite a polynomial g using two equations $f1 = f2 = 0$ that include relevant variables, say,

$$g = x^2y^3 + xy^3 + x$$

and

$$f_1 = xy - y = 0, f_2 = y^2 - x^2 = 0.$$

Naturally we assume that $f1$ and $f2$ define the following rules of rewriting:

$$xy \to y, y^2 \to x^2$$

Let us investigate the following polynomials.

$$h_1 = g - (y)f_1 = x2y3 + xy3 - xy2 + x + y2,$$

$$h_2 = g - (xy^2)f_1 = -5x + 3xy^2 + 2xy^3,$$

In these polynomials, a monomial of g is deleted and replaced by the terms according to the rules of rewriting prescribed by f_1 and f_2.

As we assume that $f_1 = 0$, this leads to that g can be reduced (or rewritten) into two equivalent expressions:

$$g \xrightarrow{f_1} h_1$$

and

$$g \xrightarrow{f_1} h_2$$

The difference between h_1 and $h2$ resulted from the positions of the deleted terms. In h_1, the first term of g is deleted, while in h_2, the second term is. To avoid this kind of ambiguity, all possible terms should be removed from the repeated reductions. This computation should be carried out in a loop while it is active until there is no possible reduction, because a reduction might yield terms that are deleted by forthcoming reductions. Let us use the expression reduce(p, q) to express this type of exhaustive reduction of a polynomial p by g. The exhaustive reductions of g by f_1 and f_2 are as follows.

$$\text{reduce}(g, f_1) = 4x^2 + 28y^2 + 12y$$

$$\text{reduce}(g, f_2) = 4x^2 - xy^2 + 4xy + x + y^2 + y$$

As we assume $f1 = f2 = 0$, we should perform secondary exhaustive reductions with different polynomials.

$$\text{reduce}(\text{reduce}(g, f_1), f_2) = 4x^2 + 28y^2 + 12y$$

$$\text{reduce}(\text{reduce}(g, f_2), f_1) = 4x^2 + x - 2y^2 + 13y$$

We observe that reduce(reduce$(g, f_1), f_2)$ and reduce(reduce$(g, f_2), f_1)$ differ. This means that the successive exhaustive reductions might result in different results if they choose different paths.

Let us translate the above consideration into a rewriting of the quantum circuit. First, assume that the polynomial g is a generator of a quantum circuit that performs the following unitary transformations:

$$U_g = \exp(ix^2y^3)\exp(ixy^3)\exp(ix)$$

where we assume $[x, y] = 0$ for simplicity, so that the splitting of the exponential operator is allowed. If the relations $f_1 = f_2 = 0$ hold, the operation U_g can be reduced to two representations.

$$U_{r_1} = \exp(4ix^2)\exp(28iy^2)\exp(i12y)$$

$$U_{r_2} = \exp(-4ix^2)\exp(ix)\exp(-i2y^2)\exp(i13y)$$

This sort of ambiguity may yield problems in quantum circuit optimization through symbolic rewriting of circuit operations: we might arrive at different results through different paths, but we cannot proceed any further, and the optimization will be suspended midway. It shall be better to obtain a unique representation after rewriting, and there arises the importance of a standard basis that enables us to obtain unique results in rewriting.

3.2 Necessity of standard (Gröbner) basis) in rewriting

Gröbner basis theory

Gröbner basis theory is a fundamental tool in computational algebraic geometry and commutative algebra, providing algorithmic methods for rewriting algebraic items.

Polynomial Rings and Ideals

Let k be a field and consider the polynomial ring $k[x_1, x_2, \ldots, x_n]$. An **ideal** $I \subseteq k[x_1, \ldots, x_n]$ is a set of polynomials closed under addition and multiplication by arbitrary polynomials in the ring.

Monomial Ordering

The computation of Gröbner bases depends fundamentally on the choice of a *monomial ordering*. A monomial ordering determines the leading term of each polynomial, and consequently affects the shape, complexity, and computational behavior of the resulting Gröbner basis. This document provides a technical overview of the global monomial orderings commonly used in computer algebra systems such as SINGULAR, based on the classification presented in the SINGULAR manual.

Let $k[x_1, \ldots, x_n]$ be a polynomial ring over a field k, and let $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ denote a monomial with multi-index $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$.

A monomial ordering $<$ is a total order on $\mathbb{N}^n$ satisfying:

1. $<$ is a well-ordering,
2. $\alpha < \beta \implies \alpha + \gamma < \beta + \gamma$ for all $\gamma \in \mathbb{N}^n$.

Global orderings are those for which $x^\alpha \to \infty$ as $|\alpha| \to \infty$. These are the standard orderings used in Gröbner basis computations.

Lexicographic Ordering (lex)

The *lexicographic ordering* compares exponent vectors from left to right. For $\alpha, \beta \in \mathbb{N}^n$, we define:

$$\alpha <_{\text{lex}} \beta \quad \iff \quad \text{for the first } i \text{ with } \alpha_i \neq \beta_i,\ \alpha_i < \beta_i.$$

- Strongly prioritizes the first variable.
- Suitable for elimination theory.
- Often computationally expensive.

Reverse Lexicographic Ordering (revlex)

The *reverse lexicographic ordering* compares exponent vectors from right to left:

$$\alpha <_{\text{revlex}} \beta \quad \iff \quad \text{for the last } i \text{ with } \alpha_i \neq \beta_i,\ \alpha_i > \beta_i.$$

- Favors monomials with larger exponents in
- Typically faster for Gröbner basis computation.
- Widely used as a default ordering.

Degree Lexicographic Ordering (deglex)

The *degree lexicographic ordering* first compares total degrees:

$$|\alpha| = \sum_{i=1}^{n} \alpha_i.$$

Then:

$$\alpha <_{\text{deglex}} \beta \iff \begin{cases} |\alpha| < |\beta|, & \text{or} \\ |\alpha| = |\beta| \text{ and } \alpha <_{\text{lex}} \beta. \end{cases}$$

- Prioritizes lower total degree.
- Behaves similarly to lex within each degree.

3.2.1 Degree Reverse Lexicographic Ordering (degrevlex)

The *degree reverse lexicographic ordering* is defined by:

$$\alpha <_{\text{degrevlex}} \beta \iff \begin{cases} |\alpha| < |\beta|, & \text{or} \\ |\alpha| = |\beta| \text{ and } \alpha <_{\text{revlex}} \beta. \end{cases}$$

- Often the most efficient ordering for Gröbner basis computation.
- Produces well-behaved initial ideals in algebraic geometry.

Weighted Orderings

Let $w = (w_1, \ldots, w_n) \in \mathbb{R}^n_{>0}$ be a weight vector. Define the weighted degree:

$$\deg_w(x^\alpha) = w_1\alpha_1 + \cdots + w_n\alpha_n.$$

Weighted Degree Lexicographic Ordering (Wp)

$$\alpha <_{\text{Wp}} \beta \iff \begin{cases} \deg_w(\alpha) < \deg_w(\beta), & \text{or} \\ \deg_w(\alpha) = \deg_w(\beta) \text{ and } \alpha <_{\text{lex}} \beta. \end{cases}$$

Weighted Degree Reverse Lexicographic Ordering (wp)

$$\alpha <_{\text{wp}} \beta \iff \begin{cases} \deg_w(\alpha) < \deg_w(\beta), & \text{or} \\ \deg_w(\alpha) = \deg_w(\beta) \text{ and } \alpha <_{\text{revlex}} \beta. \end{cases}$$

Properties

- Useful for emphasizing certain variables.
- Appears naturally in weighted projective geometry.

Comparison of Orderings

Ordering	Primary Criterion	Secondary Criterion
lex	lexicographic	none
revlex	reverse lexicographic	none
deglex	total degree	lex
degrevlex	total degree	revlex
weighted lex	weighted degree	lex
weighted revlex	weighted degree	revlex

Leading Terms and Reduction

Given a polynomial $f \in k[x_1, \ldots, x_n]$, its **leading term** $\mathrm{LT}(f)$ is the term with the highest monomial under $\prec$. Polynomial division and reduction are defined with respect to these leading terms.

Algorithm 1 REDUCE (Reduction of a Polynomial f by a Set G)

Require: Polynomial f, set of polynomials $G = \{g_1, g_2, \ldots, g_s\}$, monomial order
Ensure: Reduced form r of f modulo G

$r \leftarrow f$
while there exists $g \in G$ such that $\mathrm{LT}(g)$ divides a term in r **do**
  Choose such a $g \in G$
  Let t be a term in r divisible by $\mathrm{LT}(g)$
  Compute $m \leftarrow \frac{t}{\mathrm{LT}(g)}$
  $r \leftarrow r - m \cdot g$
end while
return r

S-Polynomials

For two polynomials $f, g \in k[x_1, \ldots, x_n]$, the **S-polynomial** is defined as:

$$S(f,g) = \frac{\mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(f)} \cdot f - \frac{\mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(g)} \cdot g$$

where $\mathrm{LM}(f)$ is the leading monomial and $\mathrm{LT}(f)$ is the leading term (coefficient included).

The Definition of the Gröbner Basis

A finite set $G = \{g_1, \ldots, g_t\} \subset I$ is a **Gröbner basis** of the ideal I if:

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle$$

That is, the ideal generated by the leading terms of G equals the ideal of leading terms of all elements in I.

Buchberger's Criterion

A set G is a Gröbner basis if and only if for all $f, g \in G$, the remainder of $S(f, g)$ upon division by G is zero.

Algorithm 2 GBASIS (Buchberger's Algorithm for Computing a Gröbner Basis)

Require: A finite set of polynomials $F = \{f_1, f_2, \ldots, f_s\}$ in $k[x_1, \ldots, x_n]$
Ensure: A Gröbner basis G for the ideal $\langle F \rangle$

```
G ← F
P ← {(f, g) | f, g ∈ G, f ≠ g}
while P ≠ ∅ do
    Choose and remove a pair (f, g) from P
    S ← S-polynomial(f, g)
    r ← Reduce(S, G)
    if r ≠ 0 then
        G ← G ∪ {r}
        for all h ∈ G \ {r} do
            P ← P ∪ {(h, r)}
        end for
    end if
end while
return G
```

Product and Chain Criteria

To improve the efficiency of Buchberger's algorithm, certain criteria are used to avoid computing unnecessary S-polynomials. There are two important criteria: the **Product Criterion** and the **Chain Criterion**.

Product Criterion

Let $f, g \in k[x_1, \ldots, x_n]$ be two polynomials in a Gröbner basis candidate set. If the leading monomials $\mathrm{LM}(f)$ and $\mathrm{LM}(g)$ are relatively prime, i.e.,

$$\gcd(\mathrm{LM}(f), \mathrm{LM}(g)) = 1,$$

then the S-polynomial $S(f, g)$ reduces to zero modulo the current Gröbner basis during the computation. Hence, the computation of $S(f, g)$ can be skipped.

Chain Criterion

Let $f, g, h \in k[x_1, \ldots, x_n]$ be polynomials such that:

$$\mathrm{LM}(h) \mid \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g)),$$

and both $S(f, h)$ and $S(g, h)$ reduce to zero modulo the current basis. Then, the S-polynomial $S(f, g)$ also reduces to zero, and its computation shall be omitted.

These criteria are useful in reducing the number of S-polynomials and significantly improve the performance of Gröbner basis computations.

3.2.2 Reduced Gröbner basis

Let $I \subset k[x_1, \ldots, x_n]$ be an ideal with a fixed monomial order. A Gröbner basis $G = \{g_1, \ldots, g_s\}$ of I is called *reduced* if:

1. Each polynomial is monic:

$$\mathrm{LC}(g_i) = 1 \quad \text{for all } i.$$

2. No term of g_i is divisible by the leading term of any other g_j:

$$\text{For } i \neq j, \text{ no monomial of } g_i \text{ is divisible by } \mathrm{LT}(g_j).$$

Theorem. For a fixed monomial order, every ideal I has a *unique* reduced Gröbner basis.

In the following, we will use the reduced Grönber basis in rewriting algebraic representations uniquely.

3.3 Summary: Letterplace Correspondence

3.3.1 The treatment of non-commutative problems in the commutative rings

In the rewriting of quantum operators, we have to consider the non-commutativity of operators, and there is an extension to the Gröbner basis theory. We will review one of these approaches in this section.

The **letterplace correspondence** is a structural embedding that translates elements of a free associative algebra—where variables do not commute—into a commutative polynomial ring with double-indexed variables. This transformation enables the application of commutative Gröbner basis techniques to non-commutative problems.

Formally, given a free associative algebra $K\langle X\rangle$ over a field K, the letterplace embedding is a map

$$\iota : K\langle X\rangle \longrightarrow K[X \mid \mathbb{N}],$$

where $K[X \mid \mathbb{N}]$ is the commutative polynomial ring in variables $(x_i \mid j)$, with $x_i \in X$ and $j \in \mathbb{N}$ representing the position (or "place") of the letter in a monomial.

For a monomial $w = x_{i_1}x_{i_2}\cdots x_{i_n} \in K\langle X\rangle$, the embedding is defined by

$$\iota(w) = (x_{i_1} \mid 0)(x_{i_2} \mid 1)\cdots(x_{i_n} \mid n-1).$$

This correspondence preserves the graded structure and allows for a one-to-one mapping between graded two-sided ideals in $K\langle X\rangle$ and a special class of shift-invariant, place-multigraded ideals in $K[X \mid \mathbb{N}]$, known as *letterplace ideals*.

By working within the commutative setting and exploiting the symmetry induced by the shift action on places, one can compute non-commutative Gröbner bases using standard commutative algorithms. The results can then be translated back to the non-commutative setting via the inverse map ι^{-1}, yielding a powerful and portable method for symbolic computation in non-commutative algebra.

A graded ideal $J \subset K[X \mid \mathbb{N}]$ is called a *letterplace ideal* if it is generated by the set

$$\bigcup_{s,d\in\mathbb{N}} s \cdot (J_d \cap V),$$

where J_d denotes the homogeneous component of J of total degree d, V is the image of the embedding $\iota : K\langle X\rangle \to K[X \mid \mathbb{N}]$ defined above. $s \cdot f$ denotes the shift action on monomials:

$$s \cdot (x_i \mid j) = (x_i \mid j + s).$$

3.3.2 The Gröbner basis of the Letterplace ideal and related algorithms

The Gröbner basis of a letterplace ideal is computed via the letterplace correspondence in an extended commutative ring. In what follows, we present the modifications of key algorithms – such as the Gröbner basis computation for the shift basis and the construction of non-commutative Gröbner basis – so that they work within the framework of letterplace ideals, following [6]. The algorithms in the commutative ring (REDUCE, GBASE, and S(f, g)) defined in the previous section are used.

Algorithm 3 SGBASIS (The Gröbner basis of the set of polynomials, including the ones generated by the shift operator)

Require: $H \subset K[X \mid \mathbb{N}]$, a finite subset such that $J = \langle s \cdot H \mid s \in \mathbb{N} \rangle$ is a shift-invariant ideal

Ensure: G, a Gröbner basis of J

$G \leftarrow H \setminus \{0\}$
$P \leftarrow \{(f, g) \mid f, g \in G, f \neq g, \gcd(\mathrm{lm}(f), \mathrm{lm}(g)) \neq 1\}$
while $P \neq \emptyset$ **do**
  Choose $(f, g) \in P$
  $P \leftarrow P \setminus \{(f, g)\}$
  $h \leftarrow \mathrm{REDUCE}(S(f, g), \bigcup_{s \in \mathbb{N}} s \cdot G)$
  if $h \neq 0$ **then**
    $G \leftarrow G \cup \{h\}$
    $P \leftarrow P \cup \{(h, s \cdot g') \mid g' \in G, s \in \mathbb{N} \gcd(\mathrm{lm}(h), \mathrm{lm}(s \cdot g')) \neq 1\}$
    $P \leftarrow P \cup \{(s \cdot h, g') \mid g' \in G, s \in \mathbb{N} \gcd(\mathrm{lm}(s \cdot h), \mathrm{lm}(g')) \neq 1\}$
  end if
end while
return G

Algorithm 4 NCGBASIS (Non-Commutative Gröbner Basis in the Letterplace framework)

Require: $F \subset K\langle X \rangle$, a finite set of homogeneous elements

Ensure: $G \subset K\langle X \rangle$, a homogeneous Gröbner basis of the two-sided ideal $\langle F \rangle$

Compute $F' := \iota(F) \subset K[X \mid \mathbb{N}]$
Compute $H := \mathrm{SGBASIS}(F')$ in $K[X \mid \mathbb{N}]$
$G' := H \cap V$ // extract elements in the image of ι
$G := \iota^{-1}(G')$
return G

3.3.3 A tool for computing Letterplace and truncated non-commutative Gröbner bases

Singular's LETTERPLACE subsystem provides computations in free associative algebras and their two-sided ideals via a commutative letterplace ring with a degree (length) bound, supporting Gröbner(-Shirshov) bases and normal forms in the truncated setting [18, 19].

4 Application of Letterplace ideal to quantum circuit rewriting

The basic way is as follows.

- Prepare equivalence relations of the gates by an ideal $J = \{fi_{=}0\}$. Each f_i should be made of two terms: a head and a tail.
- Generate the Gröbner basis G of J. The entries of the Gröbner basis $\{g_i\}$ determine the rule of rewriting: namely, if a head part of g_i is found in the target of rewriting, it should be replaced by the tail of g_i.

4.1 Examples

We study several simple cases of quantum circuit optimizations using the Letterplace ideal theory in this section.

Example 1

The polynomials used in this example are taken from [10].

We study gate operations composed of two types of gates, A and B, under two equivalence relations:

$$AAA = BBB \quad \text{and} \quad ABA = BAB.$$

These equivalences can be expressed by the following polynomials:

$$p_1 = AAA - BBB, \quad p_2 = ABA - BAB.$$

We work in the polynomial ring $K[A(1), B(1), \ldots, A(5), B(5)]$, and define the following polynomials:

$$\begin{aligned}
f_1 &= A(1)A(2)A(3) - B(1)B(2)B(3),\\
f_2 &= A(2)A(3)A(4) - B(2)B(3)B(4) = s(f_1),\\
f_3 &= A(3)A(4)A(5) - B(3)B(4)B(5) = s(s(f_1)),\\
f_4 &= A(1)B(2)A(3) - B(1)A(2)B(3),\\
f_5 &= A(2)B(3)A(4) - B(2)A(3)B(4) = s(f_4),\\
f_6 &= A(3)B(4)A(5) - B(3)A(4)B(5) = s(s(f_4)).
\end{aligned}$$

These polynomials can be interpreted as rewriting rules for gate operations. For example:

$$
\begin{aligned}
&A(1)A(2)A(3)?(4)?(5) \to B(1)B(2)B(3)?(4)?(5),\\
&?(1)A(2)A(3)A(4)?(5) \to ?(1)B(2)B(3)B(4)?(5),\\
&?(1)?(2)A(3)A(4)A(5) \to ?(1)?(2)B(3)B(4)B(5),\\
&A(1)B(2)A(3)?(4)?(5) \to B(1)A(2)B(3)?(4)?(5),\\
&?(1)A(2)B(3)A(4)?(5) \to ?(1)B(2)A(3)B(4)?(5),\\
&?(1)?(2)A(3)B(4)A(5) \to ?(1)?(2)B(3)A(4)B(5).
\end{aligned}
$$

Here, the symbol "?" denotes arbitrary gate operations that remain unchanged by rewriting/reduction.

As discussed in [10], we only need to consider the following five pairs to compute S-polynomials:

$$P = \{(f_1, f_2), (f_1, f_3), (f_1, f_6), (f_3, f_4), (f_4, f_6)\}.$$

The greater part of pairs that are excluded from P become zero after reduction, according to the product or chain criterion. There are several pairs that are not eligible for the letter place correspondence. An example of such pairs is (f_1, f_5), and their s-polynomial is as follows.

$$S(f_1, f_5) = -A(4)B(1)B(2)B(3)^2 + A(1)A(3)^2B(2)B(4)$$

In this polynomial, the place (3) is occupied twice by $B(3)^2$ (in the first term) or $A(3)^3$ (in the second term). It violates the letterplace correspondence, and the pair $S(f_1, f_5)$ is rejected.

Let $H = \{f_1, f_2, ..., f_6\}$; this is the current Gröbner basis. (We omit polynomials made by the shift.) We assume the degree reverse monomial ordering with A¿B, in other words, A(1)¿A(2)¿...¿A(5)¿B(1)¿B(2)¿...¿B(5).

Only two pairs in P remain unchanged after reduction

$$
\begin{aligned}
g_1 &= \mathrm{spoly}(f_1, f_2) = A(1)B(2)B(3)B(4) - B(1)B(2)B(3)A(4),\\
g_2 &= \mathrm{spoly}(f_4, f_6) = A(1)B(2)B(3)A(4)B(5) - B(1)A(2)B(3)B(4)A(5),
\end{aligned}
$$

and the others become zero.

Polynomials g_1 and g_2 yield two new rewriting rules:

$$
\begin{aligned}
&A(1)B(2)B(3)B(4)?(5) \to B(1)B(2)B(3)A(4)?(5),\\
&A(1)B(2)B(3)A(4)B(5) \to B(1)A(2)B(3)B(4)A(5).
\end{aligned}
$$

We extend H as follows:

$$H \to \{f_1, f_2, ..., f_6, g_1, g_2\}.$$

As all S-polynomials of H can be discarded, and we find that H is a Gröbner basis. By tracing back through the map i, we obtain the following polynomials:

$$J[1] = ABA - BAB,$$
$$J[2] = AAA - BBB,$$
$$J[3] = BBBA - ABBB,$$
$$J[4] = BABBA - ABBAB.$$

Example 2

Let us study the following polynomial:

$$p = ABA - BAB.$$

Then:

$$f_1 = i(p) = A(1)B(2)A(3) - B(1)A(2)B(3),$$
$$f_2 = s(f_1) = A(2)B(3)A(4) - B(2)A(3)B(4),$$
$$f_3 = s(s(f_1)) = A(3)B(4)A(5) - B(3)A(4)B(5).$$

We observe the following.

$$\text{spoly}(f_1, f_2) \text{ is trivial after reduction, since } \gcd(\text{lcm}(f_1), \text{lcm}(f_2)) = 1.$$

$$f_4 = \text{spoly}(f_1, f_3) = f_1 \cdot B(4)A(5) - A(1)B(2) \cdot f_3 = -B(1)A(2)B(3)B(4)A(5) + A(1)B(2)B(3)A(4)B(5).$$

Also,

$$\text{spoly}(f_2, f_3) \text{ is trivial after reduction, since } \gcd(\text{lcm}(f_2), \text{lcm}(f_3)) = 1.$$

$$\text{Reduce}(f_4, \{f_1, f_2, f_3\}) = f_4.$$

Thus, $\{f_1, f_4\}$ is a Gröbner basis of $\{f_1\}$ in $K[N|P]$, and we obtain:

$$J_1[1] = ABA - BAB,$$
$$J_1[2] = BABBA - ABBAB.$$

Example 3

Let us study the following polynomial:

$$p_1 = ABA - B.$$

We homogenize p_1 as

$$p_1 = ABA - AII,$$

where I denotes a blank symbol. We also introduce the following auxiliary relations:

$$p_2 = AA - II,$$
$$p_3 = BB - II,$$
$$p_4 = AI - IA,$$
$$p_5 = BI - IB.$$

We assume the degree reverse lexicographical monomial orderings with A¿B¿I. The Letterplace Gröbner basis of $\{p_1, p_2, p_3, p_4, p_5\}$ is:

$$J[1] = IB - BI,$$
$$J[2] = BB - II,$$
$$J[3] = IA - AI,$$
$$J[4] = AA - II,$$
$$J[5] = ABA - BII,$$
$$J[6] = BAII - ABII,$$
$$J[7] = BABII - AIIII.$$

Let us verify part of the computation.
Various polynomials are derived from the set $\{i(p_1), i(p_2), i(p_3), i(p_4), i(p_5)\}$:

$$f_1 = i(p_1) = A(1)B(2)A(3) - B(1)I(2)I(3),$$
$$f_2 = s(f_1) = A(2)B(3)A(4) - B(2)I(3)I(4),$$
$$f_3 = s(s(f_1)) = A(3)B(4)A(5) - B(3)I(4)I(5),$$
$$f_4 = i(p_2) = A(1)A(2) - I(1)I(2),$$
$$f_5 = s(f_4) = A(2)A(3) - I(2)I(3),$$
$$f_6 = s(s(f_4)) = A(3)A(4) - I(3)I(4).$$

Now consider the S-polynomial:

$$f_7 = \mathrm{spoly}(f_1, f_6) = f_1 \cdot A(4) - A(1)B(2) \cdot f_6.$$

Computing this, we obtain:

$$f_7 = B(1)I(2)I(3)A(4) - A(1)B(2)I(3)I(4).$$

It is reduced to:

$$f_7 \to B(1)A(2)I(3)I(4) - A(1)B(2)I(3)I(4).$$

This corresponds to the commutator:

$$BA - AB = 0,$$

which expresses the commutativity of A and B.

Other examples

In the Supplementary part of this article, we demonstrate some applications of the method explained herein.

5 Discussion

In this section, we examine several issues that arise when applying the algebraic approach to quantum circuit optimization.

Pruning rewriting rules

To reduce the computational cost of generating Letterplace ideals, it is necessary to employ careful strategies that prevent the system size from growing excessively.

When preparing a large collection of rewriting rules in advance, it may turn out that only a subset of them is actually effective for optimizing a given circuit. In such cases, the ineffective rules should be pruned, and only the useful ones should be retained for generating the Letterplace ideal. This selective approach can significantly reduce the computational cost of constructing the Letterplace ideals.

The existence of another rule generated by qubit permutation

When selecting a set of rewriting rules for optimizing quantum circuits, we must also consider the additional rules obtained by permuting qubits. Although it is desirable to explore these permutation-generated variants, exhaustively trying all possible qubit permutations is inefficient. To accelerate this process, we can employ techniques such as subgraph-matching algorithms (as used in [2]) to identify suitable qubit permutations more effectively.

Moreover, a set of rewriting rules and another obtained from it by permuting qubits can sometimes be unified. When this happens, the Letterplace Gröbner basis must be recomputed for the merged rule set if strict correctness is required, even though this recomputation can be time-consuming. To reduce the resulting computational cost, one can apply the product or chain criteria to determine whether the two rules interact—leading to a more complex Gröbner basis—or are independent, in which case they may be applied separately.

An extension to the Letter place ideal theory

According to the issues discussed above, the extension of the Letter place ideal algorithms suitable for optimizing quantum circuits is as follows.

Let $\{x_{t_i,q_i}\}_i$ be the set of gate operations, with type t_i operationg on qubit(s) q_i. The extended letterplace correspondence

For a monomial (representing the gate operations) $w = x_{t_1,q_1}x_{t_2,q_2}\cdots x_{t_n,q_n} \in K\langle X\rangle$, the embedding is defined by

$$\iota(w) = (x_{t_1,q_1} \mid 0)(x_{t_2,q_2} \mid 1)\cdots(x_{t_n,q_n} \mid n-1).$$

There are two operators that shift the positions of gate operations.

- The shift of places in the sequence:

  $$s(x_{t_1,q_1} \mid 0)(x_{t_2,q_2} \mid 1)\cdots(x_{t_M,q_M} \mid M{-}1)) = x_{t_1,q_1} \mid s{+}0)(x_{t_2,q_2} \mid s{+}1)\cdots(x_{t_M,q_M} \mid s{+}M{-}1)$$

  It is the same as the normal definition of the Letterplace ideal theory.

- The permutation of the qubits on which the operators act.

  $$p(x_{t_1,q_1} \mid 0)(x_{t_2,q_2} \mid 1)\cdots(x_{t_n,q_n} \mid n{-}1)) = x_{t_1,p(q_1)} \mid 0)(x_{t_2,p(q_2)} \mid M{+}1)\cdots(x_{t_M,p(q_M)} \mid M{-}1)$$

  The permutation of n qubits is the element in S_n and causes the change of the qubits on which the operators act: $p(\{q_1, q_2, ..., q_l\}) = \{q'_1, q'_2 ..., q'_l\}$ through the permutation

  $$p = \begin{pmatrix} q_1 & q_2 & \cdots & q_\ell \\ q'_1 & q'_2 & \cdots & q'_\ell \end{pmatrix}$$

The Gröbner basis computation in the commutative rings should take into account the permutation of qubits as well as the shift of places where the symbols are placed.

Let us denote the shift s and the permutation p simultaneously operated on a sequence of qubit operations g by

$$g \to (s, p) \cdot g.$$

Using these types of maps, we extend the Gröbner basis computations in the commutative rings in the Letterplace framework, as shown in the algorithm *SPGBasis*.

Algorithm 5 SPGBasis

Require: $H \subset K[X \mid \mathbb{N}]$, a finite subset such that $J = \langle (s,p) \cdot H \mid s \in \mathbb{N}, p \in S_n \rangle$ is a shift-invariant ideal

Ensure: G, a Gröbner basis of J

1: $G \leftarrow H \setminus \{0\}$
2: $P \leftarrow \{(f,g) \mid f,g \in G, f \neq g, \gcd(\mathrm{lm}(f), \mathrm{lm}(g)) \neq 1\}$
3: **while** $P \neq \emptyset$ **do**
4: Choose $(f,g) \in P$
5: $P \leftarrow P \setminus \{(f,g)\}$
6: $h \leftarrow \mathrm{REDUCE}(S(f,g), \bigcup_{s\in\mathbb{N}, p\in S_n} (s,p) \cdot G)$
7: **if** $h \neq 0$ **then**
8: $G \leftarrow G \cup \{h\}$
9: $P \leftarrow P \cup \{(h, (s,p) \cdot g') \mid g' \in G, \gcd(\mathrm{lm}(h), \mathrm{lm}((s,p) \cdot g')) \neq 1\}$
10: $P \leftarrow P \cup \{((s,p) \cdot h, g') \mid g' \in G, \gcd(\mathrm{lm}((s,p) \cdot h), \mathrm{lm}((s,p) \cdot g')) \neq 1\}$
11: **end if**
12: **end while**
13: **return** G

The heuristic approach (which produces the rules effective in rewriting) for circuit optimization is described bellow, as shown in the algorithm *BuchbergerOptimizaion*.

Algorithm 6 BuchbergerOptimizaion

Require: $H \subset K[X \mid \mathbb{N}]$, a finite subset such that $J = \langle s \cdot H \mid s \in \mathbb{N} \rangle$ is a shift-invariant ideal. This ideal represents the equivalence relations of the gate operations.

Require: $\mathcal{C}$ is the quantum circuit that is the target of the optimization.

Require: Let $T[f, \mathcal{C}]$ be the result of the opimization of $\mathcal{C}$ using the equivalence rule f.

Ensure: G, a Gröbner basis of J

$G \leftarrow H \setminus \{0\}$
$P \leftarrow \{(f, g) \mid f, g \in G, f \neq g, \gcd(\mathrm{lm}(f), \mathrm{lm}(g)) \neq 1\}$
while $P \neq \emptyset$ **do**
  Choose $(f, g) \in P$
  $P \leftarrow P \setminus \{(f, g)\}$
  $h \leftarrow \mathrm{REDUCE}(S(f, g), \bigcup_{s \in \mathbb{N}} s \cdot G)$
  if $h \neq 0$ and $T[f, \mathcal{C}] \neq \mathcal{C}$ and $T[g, \mathcal{C}] \neq \mathcal{C}$ **then**
    $G \leftarrow G \cup \{h\}$
    $P \leftarrow P \cup \{(h, g') \mid g' \in G, \gcd(\mathrm{lm}(h), \mathrm{lm}(g')) \neq 1\}$
    $\mathcal{C} \leftarrow T[h, \mathcal{C}]$
  end if
end while
return G

The large computational complexity

The difficulty of computing Letterplace ideals increases rapidly with the size of the problem. As we have seen, Gröbner basis computation in commutative polynomial rings operates in the background of the Letterplace ideal framework. The computational complexity of one of the fastest algorithms in the commutative setting, the $\mathrm{F}_5$5algorithm, is evaluated as follows.

$$N_{F_5} = \sum_{i=1}^{m} \sum_{d=\delta}^{D} b_d^{(i)} \binom{i+d-1}{d} \binom{n+d-1}{d}. \tag{1}$$

This is an estimate of the number of arithmetic operations (to be precise, the size of the data) required to compute a Gröbner basis for m homogeneous polynomials $f_1, \ldots, f_m$ in $k[x_1, \ldots, x_n]$, where k is an arbitrary field. Additionally, we assume that these polynomials are in Noether position with degrees $d_1 = \deg(f_1) \leq \cdots \leq d_m = \deg(f_m)$ and $m = n - l$ with $l \geq 0$; we define two parameters: $\delta = \delta(i) = \sum_{j=1}^{i} (d_j - 1) + 1$ and $D = \sum_{j=1}^{m} (d_j - 1) + 1$.

This estimate is based on the following idea: the F_5 algorithm decomposes the polynomial reductions in Buchberger's algorithm into a series of Gaussian eliminations on small matrices (called Macaulay matrices), whose entries are the coefficients of the polynomials. The first term on the right-hand side of the sum represents the maximum number of polynomials

reduced by Gaussian elimination on Macaulay matrices. This number grows exponentially as $m, n \to \infty$. The second and third binomial terms are the sizes of columns and rows in a Macaulay matrix, respectively, where we assume that the entries of the matrices are fully nonzero. The upper bounds of these binomials also grow exponentially as $m, n \to \infty$.

In the context of quantum circuit optimization, we have observed that every polynomial appearing in the computation consists of only two terms—the head and the tail.

$$N_{F_5}^{Qcopt} = \sum_{i=1}^{m} \sum_{d=\delta}^{D} b_d^{(i)} \binom{i+d-1}{d} \times 2 \tag{2}$$

where the sizes of rows of the Macaulay matrices are replaced by a constant 2. However, the exponential behavior of the asymptotic forms of the other terms in the summand still remains.

Consequently, deriving exact rewriting rules up to higher degrees is not always practical for large quantum circuits involving many gate operations. It is therefore advisable to alternate between two strategies for quantum circuit optimization:

1. Derive exact rewriting rules algebraically and apply them faithfully. This approach allows us to obtain a unique optimized form of a quantum circuit.

2. Use heuristic rewriting based on a collection of known circuit equivalences (without refinement via the construction of Gröbner bases), which enables exploratory, bidirectional transformations. In this case, circuit optimization may yield several distinct solutions.

6 Summary

In this article, we studied how to use the Letterplace ideal theory to optimize quantum circuits. First, we make a set of polynomials that describe the equivalences of quantum circuits, and then we generate a Letterplace ideal of this set. The set of generators of the Letterplace ideal is regarded as a collection of rewriting rules to optimize the gate operations. The merit of this approach is the uniqueness of the solutions, although the necessary algebraic computation would incur a high cost. It is advisable to combine the algebraic method presented in this article with other heuristic approaches in order to balance mathematical rigor with practical effectiveness in problem solving.

Resources

The computer programs used in the present study are available from
https://github.com/kikuchiichio/Quantum_Circuit_Optimization_by_Algebra.

References

[1] Brett Giles and Peter Selinger. Exact synthesis of multiqubit clifford+t circuits. *Physical Review A*, 2013. Accessed: 2026-01-16.

[2] Raban Iten, Romain Moyard, Tony Metger, David Sutter, and Stefan Woerner. Exact and practical pattern matching for quantum circuit optimization. *ACM Transactions on Quantum Computing*, 3(1):1–41, January 2022.

[3] Krishnageetha Karuppasamy, Varun Puram, Stevens Johnson, and Johnson P. Thomas. A comprehensive review of quantum circuit optimization: Current trends and future directions. *Quantum Reports*, 2025. Accessed: 2026-01-16.

[4] Aleks Kissinger and John van de Wetering. Reducing t-count with the zx-calculus. arXiv preprint, 2020. Accessed: 2026-01-16.

[5] Vadym Kliuchnikov. Synthesis of unitaries with clifford+t circuits. arXiv preprint, 2013. Accessed: 2026-01-16.

[6] Roberto La Scala and Viktor Levandovskyy. Letterplace ideals and non-commutative gröbner bases. *Journal of Symbolic Computation*, 44(10):1374–1393, October 2009.

[7] Roberto La Scala and Viktor Levandovskyy. Skew polynomial rings, gröbner bases and the letterplace embedding of the free associative algebra. *Journal of Symbolic Computation*, 48(1):110–131, January 2013. http://arxiv.org/abs/1009.4152.

[8] Viktor Levandovskyy, Tobias Metzlaff, and Karim Abou Zeid. Computation of free non-commutative gröbner bases over $\mathbb{Z}$ with singular:letterplace. In *Proceedings of the 2020 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 312–319. ACM, 2020. https://av.tib.eu/media/50124.

[9] Viktor Levandovskyy, Hans Schoenemann, and Karim Abou Zeid. Letterplace - a subsystem of singular for computations with free algebras via letterplace embedding. In *Proceedings of the 2020 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 305–311. ACM, 2020. https://av.tib.eu/media/50123.

[10] Viktor Levandovskyy, Grischa Studzinski, and Benjamin Schnitzler. Enhanced computations of gröbner bases in free algebras as a new application of the letterplace paradigm. In *Proceedings of the 2013 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 259–266. ACM, 2013.

[11] M. Miller et al. Quantum circuit simplification using templates. DATE paper, 2004. Accessed: 2026-01-16.

[12] Teo Mora. *Solving Polynomial Equation Systems IV: Buchberger Theory and Beyond*. Cambridge University Press, 2016.

[13] Tom Peham, Nina Brandl, Richard Kueng, Robert Wille, and Lukas Burgholzer. Depth-optimal synthesis of clifford circuits with sat solvers. arXiv preprint, 2023. Accessed: 2026-01-16.

[14] PyZX Developers. Optimizing and simplifying circuits — pyzx documentation, 2026. Accessed: 2026-01-16.

[15] PyZX Developers. zxcalc/pyzx: Python library for quantum circuit rewriting and optimisation using the zx-calculus, 2026. Accessed: 2026-01-16.

[16] Roberto La Scala. Extended letterplace correspondence for nongraded noncommutative ideals and related algorithms. *International Journal of Algebra and Computation*, 24(08):1157–1182, 2014.

[17] Leonard Schmitz and Viktor Levandovskyy. Formally verifying proofs for algebraic identities of matrices. In *Intelligent Computer Mathematics (CICM 2020)*, Lecture Notes in Artificial Intelligence (LNAI), LNCS, pages 222–236. Springer, 2020.

[18] Singular Team. Singular manual: Letterplace, 2023. Accessed: 2026-01-16.

[19] Singular Team. Singular manual: Letterplace correspondence, 2024. Accessed: 2026-01-16.

[20] Mingkuan Xu, Zikun Li, Oded Padon, Sina Lin, Jessica Pointing, Auguste Hirth, Henry Ma, Jens Palsberg, Alex Aiken, Umut A Acar, et al. Quartz: superoptimization of quantum circuits. In *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 625–640, 2022.

[21] Dekel Zak, Jingyi Mei, Jean-Marie Lagniez, and Alfons Laarman. Reducing quantum circuit synthesis to #sat. In *31st International Conference on Principles and Practice of Constraint Programming (CP 2025)*, 2025. Accessed: 2026-01-16.

A Supplemental part

We demonstrate how to apply the rewriting method presented in the main body of the article.

A.1 Monomial orderings

Let us examine how the monomial orderings alter the alignments of monomials in polynomials.

Lexicographical ordering

With $A > B > C$ in the lexicographical ordering, the polynomial $(A + B + C + 1)^3$ is expanded as follows. $A^3 + 3A^2B + 3A^2C + 3A^2 + 3AB^2 + 6ABC + 6AB + 3AC^2 + 6AC + 3A + B^3 + 3B^2C + 3B^2 + 3BC^2 + 6BC + 3B + C^3 + 3C^2 + 3C + 1$

Degree reverse lexicographical ordering

With $A > B > C$ in the degree reverse lexicographical ordering, the same polynomial is expanded as follows. $A^3 + 3A^2B + 3AB^2 + B^3 + 3A^2C + 6ABC + 3B^2C + 3AC^2 + 3BC^2 + C^3 + 3A^2 + 6AB + 3B^2 + 6AC + 6BC + 3C^2 + 3A + 3B + 3C + 1$

Weighted degree reverse lexicographical ordering

With $A > B > C$ in the weighted degree reverse lexicographical ordering and with weights $W_A = 1, W_B = 10, \text{and} W_C = 2$, the same polynomial is expanded as follows. $B^3 + 3B^2C + 3AB^2 + 3B^2 + 3BC^2 + 6ABC + 3A^2B + 6BC + 6AB + 3B + C^3 + 3AC^2 + 3A^2C + 3C^2 + A^3 + 6AC + 3A^2 + 3C + 3A + 1$

How to adjust rewriting rules by altering monomial orderings

In the rewriting of the gate operations, a polynomial stands for a rule:

$$\text{The leading monomial} \rightarrow \text{The remaining part.}$$

Consequently, the gate operations equipped with the higher orderings are replaced by the terms with lower orderings. For example, if one wants to remove Toffoli gates in the circuit optimization, one should use monomial orderings such as $ccx > cx > x$.

A.2 Deriving rewriting rules

Let us study the following four equivalent circuits.

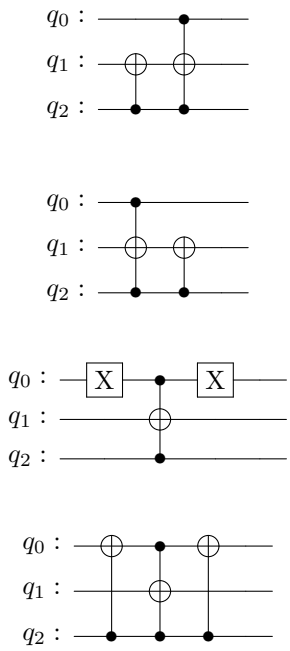

Figure 1: Equivalent Four circuits

From these circuits, we construct the following three polynomials.

$cx21{*}ccx021{-}ccx021{*}cx21, ccx021{*}cx21{-}x0{*}ccx021{*}x0, x0{*}ccx021{*}x0{-}cx20{*}ccx021{*}cx20;$

In these polynomials, gate operations are denoted by symbols such as x_0, cx_{01}, and ccx_{021}. These symbols represent the gate type (x, cx, or ccx) and the target/control qubits (indicated by integers). For example, ccx_{021} refers to a Toffoli gate with controls at q_0 and q_2, and a target X gate at q_1. Furthermore, the self-inverse property of these gates is defined by the following additional polynomials:

$$x0 * x0 - 1, cx21 * cx21 - 1, cx20 * cx20 - 1, ccx021 * ccx021 - 1.$$

```
LIB "freegb.lib";
ring r=0, (ccx021, cx20, cx21, x0),dp;

for (int bdeg=3; bdeg<=20; bdeg=bdeg+1)
{

def R = freeAlgebra(r, bdeg);
```

```
setring R;
ideal I=cx21*ccx021-ccx021*cx21,ccx021*cx21-x0*ccx021*x0,x0*ccx021*x0-cx20
↪  *ccx021*cx20;
ideal I1=x0*x0-1,cx21*cx21-1,cx20*cx20-1,ccx021*ccx021-1;
ideal J = letplaceGBasis(I+I1);
print(string(bdeg)+","+string(size(J)));
kill I;
kill I1;
kill J;
setring r;
kill R;
}
```

The LetterPlace Gröbner basis up to 5 degrees consists of twenty-three polynomials shown below. Each polynomial is made of two monomials: Head − Tail (the head and the tail), representing the rewriting rule Head → Tail if the head is found in sequences of gate operations.

```
J[1]=x0*x0-1
J[2]=x0*cx21-cx21*x0
J[3]=cx21*cx21-1
J[4]=cx21*cx20-cx20*cx21
J[5]=cx20*cx20-1
J[6]=cx21*ccx021-ccx021*cx21
J[7]=ccx021*ccx021-1
J[8]=ccx021*cx21*x0-x0*ccx021
J[9]=x0*ccx021*x0-ccx021*cx21
J[10]=ccx021*x0*cx21-x0*ccx021
J[11]=ccx021*cx20*cx21-cx20*ccx021
J[12]=x0*ccx021*cx21-ccx021*x0
J[13]=cx20*ccx021*cx21-ccx021*cx20
J[14]=cx21*x0*cx20-x0*cx20*cx21
J[15]=ccx021*cx21*cx20-cx20*ccx021
J[16]=cx20*ccx021*cx20-ccx021*cx21
J[17]=cx21*x0*ccx021-ccx021*x0
J[18]=cx20*x0*ccx021-ccx021*cx20*x0
J[19]=ccx021*x0*ccx021-cx21*x0
J[20]=x0*cx20*ccx021-ccx021*x0*cx20
J[21]=ccx021*cx20*ccx021-cx21*cx20
J[22]=ccx021*x0*cx20*cx21-x0*ccx021*cx20
```

```
J[23]=ccx021*cx20*x0*cx20*cx21-cx20*ccx021*x0*cx20
```

A.3 Product criterion: prediction of unnecessary computations

Let us consider the polynomials representing the self-inverse properties:

$$x0 * x0 - 1, cx21 * cx21 - 1, cx20 * cx20 - 1, ccx021 * ccx021 - 1;$$

This is an example where the product criterion holds. The entries of Gröbner basis of these polynomials are themselves.

A.4 Reduction/Rewriting of quantum circuits using templates

Let us study the following two circuits.

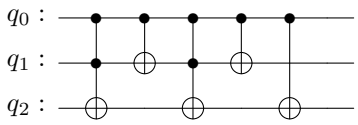

Figure 2: The circuit Q1

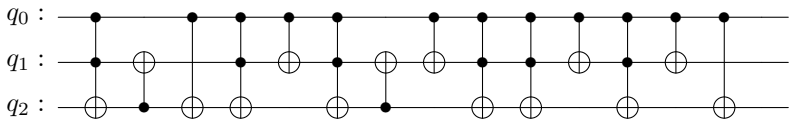

Figure 3: The circuit Q2

We try to rewrite the circuit Q1 with the condition that Q2 $= I$.

We use the following polynomials.

```
Q1=ccx012*cx01*ccx012*cx01*cx02
```

```
Q2=ccx012*cx21*cx02*ccx012*cx01*ccx012*cx21*cx01*ccx012-1
```

We use the degree reverse lexicographical ordering with ccx012¿cx21¿cx01¿cx02. We perform the calculation using the following script.

```
LIB "freegb.lib";
ring r=0,(ccx012,cx21,cx01,cx02),dp;
def R = freeAlgebra(r, 10);
```

```
setring R;
ideal I = ccx012*cx01*ccx012*cx01*cx02-1,ccx012*cx21*cx02*ccx012*cx01*ccx0
↪  12*cx21*cx01*ccx012-1;
ideal I1=ccx012*ccx012-1,cx21*cx21-1,cx01*cx01-1,cx02*cx02-1,cx01*cx02-cx0
↪  2*cx01;
ideal J = letplaceGBasis(I+I1);
ideal I = ccx012*cx21*cx02*ccx012*cx01*ccx012*cx21*cx01*ccx012-1;
ideal I1=ccx012*ccx012-1,cx21*cx21-1,cx01*cx01-1,cx02*cx02-1,cx01*cx02-cx0
↪  2*cx01;
ideal J = letplaceGBasis(I+I1);
poly Q1=ccx012*cx01*ccx012*cx01*cx02;
reduce(Q1,J);
```

In the above computation, additional rules are derived from the polynomial Q_2, the self-inverse property, and commutativity relations, considering terms up to degree 10. These rules encompass not only reduction patterns such as

$$A_1 A_2 \cdots A_n \to I \tag{3}$$

but also transformation patterns such as

$$B_1 B_2 \cdots B_l \to C_1 C_2 \cdots C_m. \tag{4}$$

We obtain $Q \to$ cx21*cx01*cx21*cx01, and its goal is illustrated in Figure 4.

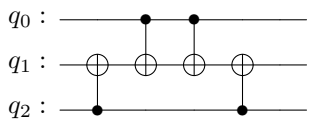

Figure 4: The result of the optimization of Q1

In truth, there is room for further optimization in this result, since we have omitted the commutativity of cx21 and cx01. We apply it to the reduction of the polynomial cx21*cx01*cx21*cx01, and we get cx21*cx21*cx01*cx01, which is apparently unity.

A.5 The necessity of qubit permutation

We discuss the importance of treating equivalent circuits properly that are made by the permutations of qubits.

Let us optimize the circuit Q1 with another template Q3, provided that Q3 = I. Q3 is a circuit made by a permutation of qubits from the circuit Q2.

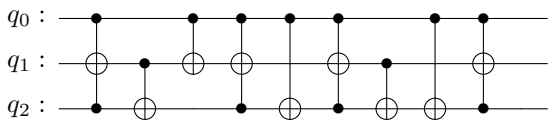

Figure 5: The circuit Q3. This circuit is made by the permutation of qubits from the circuit Q2.

The circuit Q3 is represented by a polynomial.

```
Q3=ccx021*cx12*cx01*ccx021*cx02*ccx021*cx12*cx02*ccx012
```

We use the degree reverse lexicographical ordering with ccx012¿ccx021¿cx12¿cx01¿cx02. We perform the calculation using the following script.

```
LIB "freegb.lib";
ring r=0,(ccx012,ccx021,cx12,cx01,cx02),dp;
def R = freeAlgebra(r, 10);

setring R;


ideal I1=ccx021*cx12*cx01*ccx021*cx02*ccx021*cx12*cx02*ccx012-1,ccx012*cx0⌋
↪  2-cx02*ccx012,cx01*cx02-cx02*cx01;
ideal J = letplaceGBasis(I1);
poly Q1=ccx012*cx01*ccx012*cx01*cx02;
reduce(Q1,J);
```

In the above case, several rules are derived from the polynomial Q3, the self-inverse property, and the commutativity of the gate operations, but the reduction by those rules keeps the polynomial Q1 (namely, the circuit Q1) intact. However, as we demonstrated in the previous section, Q1 is reduced to unity with the template Q2 = I, which is equivalent to Q3 = I through a permutation of the qubits. These two examples teach us that, without the proper application of templates generated by possible qubit permutations, we may miss some patterns for rewriting and fail in the optimization of quantum circuits.

A.6 Example

Let us optimize the circuit Q2 with the template Q1 = I. We use the degree reverse lexicographical ordering with ccx012¿cx21¿cx01¿cx02. We perform the calculation using

the following script.

```
LIB "freegb.lib";
ring r=0,(ccx012,cx21,cx01,cx02),dp;
def R = freeAlgebra(r, 10);

setring R;
ideal Q1 = ccx012*cx01*ccx012*cx01*cx02-1;
ideal I1=ccx012*ccx012-1,cx21*cx21-1,cx01*cx01-1,cx02*cx02-1,cx01*cx02-cx0
↪  2*cx01,cx21*cx01-cx01*cx21;
ideal J = letplaceGBasis(Q2+I1);
poly Q2=ccx012*cx21*cx02*ccx012*cx01*ccx012*cx21*cx01*ccx012;
reduce(Q2,J);
```

The reduction of Q2 results in unity.