A study on CNOT-based quantum circuits using rewriting systems

坂下, 一生

https://doi.org/10.15017/1398533

出版情報：九州大学, 2013, 博士（機能数理学）, 課程博士
バージョン：

A study on CNOT-based quantum circuits using rewriting systems

Issei Sakashita

Graduate School of Mathematics

Kyushu University

2013

Contents

1 Introduction

Rewriting techniques are very common in many area of Computer Science and related fields. Term rewriting system can be used to represent abstract interpreters of programming languages and to model formula manipulating systems used in various applications, such as program optimization, program validation, and automatic theorem proving [Hue80]. Terms are transformed by given rewriting rules. If we are not able to apply rewriting rules any more, then the computation based on rewritings is finished and we decide results of computations by the final term. Rewritings without overlapping can be done parallel, so it can be considered as a computational model of parallel computing. The confluence property is one of the most important properties of term rewriting system. Various sufficient criteria for proving this property have been widely investigated. A necessary and sufficient criterion for confluence of terminating term rewriting system, in which every reduction must terminate, was demonstrated by Knuth and Bendix [KB70]. The completion procedure suggests that rewriting system is closely related to the theory of equation.

Quantum computers were proposed in the early 1980s [Ben80, Ben82]. Significant contributions to quantum algorithms include the Shor factorization algorithm [Sho94, Sho97] and the Grover search algorithm [Gro96]. The

quantum circuit model of computation is due to Deutsch [Deu89], and it was further developed by Yao [Yao93]. After the works of Deutsch and Yao the concept of a universal set of quantum gates became central in the theory of quantum computation. A set $G = \{G_{1,n_1}, \cdots, G_{r,n_r}\}$ of r quantum gates G_{j,n_j} acting on n_j qubits ($j = 1, \cdots, r$), is called universal if any unitary action U_n on n input quantum states can be decomposed into a product of successive actions of G_{j,n_j} on different subsets of the input qubits [GMD02]. A first example of 3 qubits universal gate sets consists of Deutsch's gates **Q** [Deu89]. The gate **Q** is an extension of the Toffoli gate [Tof81]. DiVincenzo showed that a set of 2 qubits gates is exactly universal for quantum computation [DiV95]. After the result of DiVincenzo, Barenco showed that a large subclass of 2 qubits gates are universal, and moreover, that almost any 2 qubits gates is universal [Bar95]. Barenco et al. showed that the set consisting of 1 qubit gates and CNOT gates is universal [BBC$^+$95]. There have been a number of studies that investigate the number of gates for decomposing any gate of n qubits in $U(2^n)$.

- $G_1 = \{U_2 : U_2 \in U(2^2)\}$ (DiVincenzo 1995)
- $G_2 = \{U_1, CNOT : U_1 \in U(2)\}$ (Barenco et al. 1995)
- $G_3 = \{D\}$ Deutch gate (Deutsch 1989)

For the universal set consisting of 1 qubit gates and CNOT gates, Barenco et. al. showed the number of gates is $O(n^3 4^n)$ [BBC$^+$95]. Knill reduced this bound to $O(n4^n)$ [Kni95]. Most useful information about universal quantum gates can be obtained from a survey paper written by A. Galindo and M.A. Marin-Delgado [GMD02].

The design of a good quantum circuit plays a key role in the successful implementation of a quantum algorithm. For this reason, Iwama et al. presented transformation rules that transform any 'proper' quantum circuit into a 'canonical' form circuit [IKY02]. There is, however, no discussion about the minimal size of a quantum circuit. In this article, we formulate a quantum circuit as a string and then simplify the circuit by using string rewriting rules to investigate them formally. Since a string rewriting system can be analyzed by using a monoid, we require several properties about monoids and groups.

String rewriting systems simplify strings by using transformation rules, and they have played a major role in the development of theoretical computer science. Several studies of string rewriting systems have been investigated [BO93]. Let M be a monoid and T a submonoid of finite index in M. If T can be presented by a finite complete rewriting system, so M can [Wan98]. The problem of confluence is, in general, undecidable. Parkes et

al. showed that the class of groups that have monoid presentations obtainable by finite special [λ]-confluent string rewriting systems strictly contains the class of plain groups (the groups which are free products of a finitely generated free group and finitely many finite groups) [PS04]. The word problem is, in general, undecidable. If R is a finite string rewriting system that are Noetherian and confluent, then the word problem is decidable [Boo82, OZ91]. Book considered the word problem for finite string rewriting systems in which the notion of 'reduction' is based on rewriting the string as a shorter string [Boo82]. He showed that for any confluent systems of this type, there is a linear-time algorithm for solving the word problem. Using a technique developed in [Boo82], Book and Ó'Dúlaing [BO81] showed that there is a polynomial-time algorithm for testing if a finite string rewriting system is confluent. Gilman [Gil79] considered a procedure that, beginning with a finite string rewriting system, attempts to construct an equivalent string rewriting system that is Noetherian and confluent, that is, a string rewriting system such that every congruence class has a unique 'irreducible' string. This procedure appears to be a modification of the completion procedure developed by Knuth and Bendix [KB70] in the setting of term-rewriting systems. Narendran and Otto [NO88] also contributed to this topic. Later, Kapur and Narendran [KN85] showed how the Knuth-Bendix completion

algorithm could be adapted to the setting of string rewriting systems.

We do not deal with the general theory whether string rewriting systems are decidable or undecidable. We introduce an idea to reduce the size of a quantum circuit by using a string rewriting system. Our string rewriting

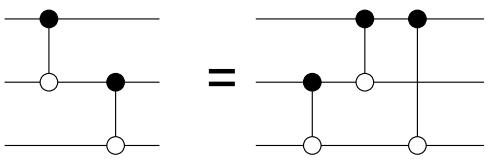

Figure 1: A circuit equation

rules based on 18 equations introduced by Iwama et al. 2002 [IKY02]. The Iwama's equations can not be considered as a complete rewriting rules as it is. That is, it does not have properties of termination and confluence. We would like to obtain a complete transformation rule set (i.e., a set of transformation rules with the properties of 'termination' and 'confluence') for reducing a quantum circuits. Therefore, we apply the Knuth-Bendix completion algorithm to a set of modified 18 equations. We obtain our complete transformation rule set consisted of 114 rules. We also obtain the length of a normal form is at most 6 and the number of normal forms is

168. Furthermore, we found a minimal subset of equations which induce the same rewriting system.

We use the Mathematica software to obtain a complete transformation rule set. We implemented the `KnuthBendix` function that obtain a complete transformation rule set, and the `Irreduce` function that obtain an irreducible transformation rule set. We investigated about the *Cayleygraph* that is constructed by quantum circuits.

This article consists of as follows. In section 2, we define several properties for term rewriting systems and string rewriting systems . In section 3, we describe formal definitions of a quantum circuit. We consider a circuit that consists of just CNOT gates on 3 qubits. We define a quantum circuit rewriting system for 3 qubits and show several related properties about it. We show that the number of normal forms is 168 on 3 qubits. We obtain a minimal subset of equations that is a good hint to construct an efficient initial equation set for n qubits. Further, we consider how to extend n qubits quantum circuits rewriting system. In section 4, we introduce our software implementations. We show examples of the implementations that compute complete transformation rule sets. Moreover, we consider the Cayley graph of 3 qubits circuits rewriting system.

2 Rewriting System

2.1 Abstract Reduction System

In this section, we introduce definitions of an abstract reduction system and properties of it.

Definition 2.1. (Abstract Reduction System) Let A be a set and $\rightarrow$ a binary relation on A. An *Abstract Reduction System* $\mathcal{A}$ is a pair $\mathcal{A} = (A, \rightarrow)$. We call the binary relation as a *reduction relation*. For $a, b \in A$, if $(a, b) \in \rightarrow$, we write $a \rightarrow b$ and call b a one-step reduct of a. The transitive reflexive closure of $\rightarrow$ is written as $\rightarrow^*$. $a \rightarrow^* b$ is defined as

$$a = a_0 \rightarrow a_1 \rightarrow \cdots \rightarrow a_n = b, \ (n \geq 0).$$

We define confluence property of rewriting system. The property is one of the most important properties of a rewriting system. If we identify the transformations with computation, the confluence property assures uniqueness of answers.

Definition 2.2. (weakly confluent) A reduction relation $\rightarrow$ is called *weakly confluent* (WCR) if $\forall\ a, b, c \in A\ (a \rightarrow b \text{ and } a \rightarrow c) \Longrightarrow \exists d \in A\ (b \rightarrow^* d \text{ and } c \rightarrow^* d)$.

Definition 2.3. (confluent) A reduction relation $\rightarrow$ is called *confluent* (CR) if $\forall\ a, b, c \in A\ (a \rightarrow^* b \text{ and } a \rightarrow^* c) \Longrightarrow \exists d \in A\ (b \rightarrow^* d \text{ and } c \rightarrow^* d)$.

We draw the difference of weakly confluent and confluent is Figure 2. Weakly confluent suppose one-step reductions of a. Confluent suppose transitive reflexive closure $\rightarrow^*$ of a.

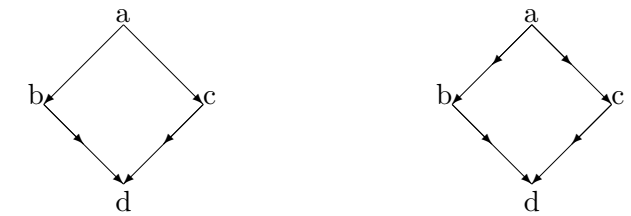

Figure 2: weakly confluent and confluent

We define termination property of rewriting system. The property is one of the most important properties of rewriting system. The termination property assures the termination of computations.

Definition 2.4. (normal form) We call that $a \in A$ is a *normal form* if there is no $b \in A$ such that $a \rightarrow b$. Further, $b \in A$ has a $normal form$ if there exist a transitive reflexive closure $\rightarrow^*$ that is $b \rightarrow^* a$ for some *normal form* $a \in A$. We denote the normal form of b as $NF(b)$.

Definition 2.5. (weakly normalizing, strongly normalizing) A reduction relation $\rightarrow$ is *weakly normalizing* if every $a \in A$ has a normal form. A reduction relation $\rightarrow$ is *strongly normalizing* (SN) if every reduction sequence $a_0 \rightarrow a_1 \rightarrow \cdots$ eventually must terminate in normal form.

Lemma 2.1. (Newman's lemma) Let $(A, \rightarrow)$ be an abstract reduction system. If $(A, \rightarrow)$ is strongly normalizing and weakly confluent, then it is confluent.

$$\text{SN } \& \text{WCR } \Longrightarrow \text{CR}$$

□

Strongly normalizing and confluent are important properties. Therefore we define a complete transformation rule set that is strongly normalizing and confluent.

Definition 2.6. (complete transform rule set) A set of rewriting rule set R is *complete* if R is SN and CR.

2.2 Term Rewriting System

In this section, we introduce definitions of term rewriting system and we summarize elementary the properties of it.

Definition 2.7. (term rewriting system) A term rewriting system (Σ, R) is a pair of an alphabet Σ and a set of rewriting rules R. The alphabet consists of:

1. A countably infinite set of *variables* x_1, x_2, x_3, $\cdots$.

2. A non-empty set of *function symbols* $F, G, \cdots$ each equipped with an *arity* (a natural number), i.e. the number of *arguments* it is supposed to have. 0-ary is a *function symbol*. We call it a *constant symbol*.

A set of terms over Σ is $Ter(\Sigma)$. It is defined inductively:

1. If $x_1, x_2, x_3, \cdots \in \Sigma$ are variables, then $x_1, x_2, x_3, \cdots \in Ter(\Sigma)$.

2. If F is an n-ary function symbol and $t_1, \cdots t_n \in Ter(\Sigma)$ $(n \geq 0)$, then $F(t_1, \cdots, t_n) \in Ter(\Sigma)$.

A set of rewriting rules $R = \{s_i \to t_i | s_i, t_i \in Ter(\Sigma), i \in I\}$ consists of:

1. s_i is not variable.

2. Let $\mathcal{V}(t)$ be a set of variables that is contained in t. $\mathcal{V}(t_i) \subseteq \mathcal{V}(s_i)$.

We define several definitions of term rewriting system to define a *critical pair*.

Definition 2.8. (Position, Subterm) For all $t \in Ter(\Sigma)$, we define a set of *positions* $\mathcal{O}(t)$ and a *subterm* of t for the *position* u. We denote the *subterm* of t for the *position* u as t/u.

1. If t is a variable, then $\mathcal{O}(t) = \{\epsilon\}$ and $t/\epsilon = t$.

2. If $t = f(t_1, \cdots, t_n)$, then

$$\mathcal{O}(t) = \{\epsilon\} \cup \{i \cdot u \mid i \leq n, u \in \mathcal{O}(t_i)\} \text{ and}$$

$$t/\epsilon = t,\ t/(i \cdot u) = (t/i)/u.$$

Example 2.1. For a term $t = plus(plus(x, y), z)$, the set of positions is

$$\mathcal{O}(t) \quad = \quad \{\epsilon, 1, 1 \cdot 1, 1 \cdot 2, 2\}.$$

A subterm $t/1$ is plus(x, y). A subterm $t/(1 \cdot 2)$ is

$$t/(1 \cdot 2) = (t/1)/2 = \text{plus(x,y)}/2 = y.$$

We draw a figure 3 that express subterms of t for all positions.

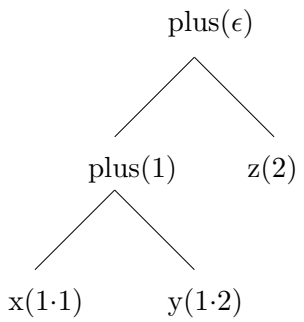

Figure 3: A example of position

Definition 2.9. (substitution) A *subtituition* is a finite set $\{v_1 \to t_1, \cdots, v_n \to t_n\}$ where every v_i is a variable and every t_i is a term. No two elements in the set haeve the same variable v_i.

Let t be a term and θ a substitution. We denote the substituting all variables of t by θ as $\theta(t)$.

Definition 2.10. (Composition of substitution) Let $\theta = \{x_1 \to t_1, \cdots, x_n \to t_n\}$ and $\lambda = \{y_1 \to u_1, \cdots y_m \to u_m\}$ be substitutions. Then the *composition* of θ and λ is defined by building the set

$$\{x_1 \to \lambda(t_1), \cdots, x_n \to \lambda(t_n), y_1 \to u_1, \cdots, y_m \to u_m\},$$

and deleting the following elements

- any element $x_j \to \lambda(t_j)$ such that $\lambda(t_j) = x_j$ and
- any element $y_j \to u_j$ such that y_j is in $\{x_1, \cdots, x_n\}$.

We denote the *composition* of θ and λ as $\theta \circ \lambda$.

Definition 2.11. (unifier)

Let θ be a substitution and t_1, t_2, $\cdots$, and t_n terms. θ is called a unifier for a set $\{t_1, \cdots, t_n\}$ if and only if

$$\theta(t_1) = \theta(t_2) = \cdots = \theta(t_n).$$

The set $\{t_1, \cdots, t_n\}$ is unifiable if and only if there exists a unifier for it.

Definition 2.12. (most general unifier)

Let θ be a substitution and t_i $(i = 1, \cdots, n)$ terms. θ is called a most general

unifier for a set $\{t_1, \cdots, t_n\}$ if and only if for any unifier ν there exists a substitution γ such that $\nu = \theta \circ \gamma$.

Definition 2.13. (overlap of terms)

Let $r_1 : s_1 \to t_1$ and $r_2 : s_2 \to t_2$ be rewriting rules. If there exists a position $u \in \mathcal{O}(s_1)$ and an unifier θ such that

$$s_1/u \notin \mathcal{V} \text{ and } \theta(s_1/u) \equiv \theta(s_2)$$

then we call that r_2 *overlap* with r_1.

There is a possibility that reducing terms occurs *critical pairs*. Let s, t be terms and the position $u \in \mathcal{O}(t)$. $t[u \leftarrow s]$ is a term rewritten a subterm t/u to s.

Definition 2.14. (Critical pair)

Let $r_1 : s_1 \to t_1$, $r_2 : s_2 \to t_2$ be rewriting rules that r_2 overlap with r_1 at the position $u \in \mathcal{O}(s_1)$ by most general unifier θ. A critical pair of r_1 and r_2 is a pair of terms such that

$$\langle \theta(s_1[u \leftarrow t_2]), \theta(t_1) \rangle$$

Example 2.2. Let r_1, r_2 rewriting rules such that

$$\begin{aligned} r_1 &: (x' + y') + z' \to x' + (y' + z') \\ r_2 &: (-x) + x \to 0, \end{aligned}$$

and θ a most general unifier such that

$$\theta(x') = -x,\ \theta(y') = x.$$

r_2 overlap with r_1 at the position (1). We compute a critical pair of r_1 and r_2.

$$\theta(s_1[u \leftarrow t_2]) = 0 + z'$$

$$\theta(t_1) = (-x) + (x + z')$$

Therefore,

$$\langle \theta_{s_1}[u \leftarrow t_2], \theta_{t_1} \rangle = \langle 0 + z', (-x) + (x + z') \rangle.$$

Lemma 2.2. Let (Σ, R) be a term rewriting system, CP a set of critical pairs for R.

(Σ, R) is WCR if and only if $\forall \langle p, q \rangle \in CP,\ \exists d \in A(p \rightarrow^* d \text{ and } q \rightarrow^* d)$.

□

Since this lemma and Newman's lemma, we have the following proposition.

Proposition 2.1. Let (Σ, R) be a term rewriting system which reduction relation is SN. Let CP be a set of critical pairs for R.

(Σ, R) is CR if and only if $\forall \langle p, q \rangle \in CP,\ \hat{p} \equiv \hat{q}$.

We denote the normal form of t as $\hat{t}$.

□

Example 2.3. There are 3 transformation rules where

$$r_1 : 0 + x \to x$$

$$r_2 : (-x) + x \to 0$$

$$r_3 : (x + y) + z \to x + (y + z).$$

We are able to have a complete transformation rule set for the 3 rules in Table 1.

$r_1 : 0 + x \to 0$
$r_2 : (-x) + x \to 0$:
$r_3 : (x + y) + z \to x + (y + z)$
$r_4 : (-x) + (x + y) \to y$
$r_5 : x + 0 \to x$
$r_6 : x + (-x) \to 0$
$r_7 : x + ((-x) + y) \to y$
$r_8 : (-0) \to 0$
$r_9 : (-(-x)) \to x$
$r_{10} : (-(y + x)) \to (-x) + (-y)$

Table 1: An example of complete rewriting system

2.3 String Rewriting System

In this section, we introduce the definition of a string rewriting system, for investigating about quantum circuits using it. Let Σ be a finite set of alphabets. We denote the set of all strings over Σ, including the empty string λ, as Σ^*. The length of a string $w \in \Sigma^*$ is denoted by $|w|$. A rewriting rule (u, v) is a pair of strings $u, v \in \Sigma^*$ where $u \neq \lambda$.

Definition 2.15 (string rewriting system)**.** A string rewriting system is a pair (Σ, R) of a finite set of alphabets Σ and a finite set of rewriting rules R .

Definition 2.16 (string rewriting)**.** Let (Σ, R) be a rewriting system and $s, t \in \Sigma^*$. We denote $s \to_R t$ if and only if there exist strings x, y, u and v in Σ^* such that $s = xuy, t = xvy$ and $(u, v) \in R$.
The reflexive transitive closure relation of $\to_R$ over Σ^* is denoted by $\to_R^*$. Further $\leftrightarrow_R^*$ is the symmetric closure relation of $\to_R^*$.

An equivalence class of a string rewriting systems are considered using monoids, so we introduce several definitions and properties about monoids and their interprelations.

Definition 2.17 (monoid)**.** A monoid $M = (M, \cdot, \lambda)$ is a tuple of a set M, a binary operation $\cdot : M \times M \to M$, and a unit element $e \in M$ that satisfies

the following two axioms.

- For any a, b, and c in $M, (a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- For any a in M, $a \cdot \lambda = \lambda \cdot a = a$.

We note $(\Sigma^*, \cdot, \lambda)$ is a monoid where $\cdot$ is concatenation and λ is an empty string.

Definition 2.18 (homomorphism, isomorphic)**.** A homomorphism between two monoids $(M_1, \cdot_1, \lambda_1)$ and $(M_2, \cdot_2, \lambda_2)$ is a function $f : M_1 \to M_2$ such that

- $f(x \cdot_1 y) = f(x) \cdot_2 f(y)$ for any $x, y \in M_1$, and
- $f(\lambda_1) = \lambda_2$.

If there exists a bijective homomorphism $f : M_1 \to M_2$, then M_1 and M_2 are isomorphic. We denote isomorphic as $M_1 \sim M_2$.

Proposition 2.2. Let Σ be a finite set and $(M, \cdot, \lambda)$ a monoid. A function $f : \Sigma \to M$ is uniquely extended to the homomorphism $f^* : \Sigma^* \to M$ where $f^*(x_1 \cdot x_2 \cdots x_n) = f(x_1)f(x_2) \cdots f(x_n)$ and $f^*(\lambda) = \lambda$.

□

Definition 2.19 (model, interpretation)**.** Let (Σ, R) be a rewriting system and $(M, \cdot, \lambda)$ a monoid. We say $(M, \cdot, \lambda)$ is a model of (Σ, R) if there exists

a function $f : \Sigma \to M$ such that $f^*(u) = f^*(v)$ for any $(u, v) \in R$. We call the function f^* an interpretation of the string rewriting system (Σ, R) to a monoid M.

A string rewriting system can be investigated using monoids and interpretations. We now add further definitions for discussing the equivalence of rewriting systems.

Definition 2.20 (factor monoid)**.** Let (Σ, R) be a rewriting system. A factor monoid $(\Sigma^*/R, \cdot, [\lambda])$ is defined by $\Sigma^*/R = \Sigma^*/ \leftrightarrow_R^*$ and $[x] \cdot [y] = [xy]$ where $[x] = \{x' | x \leftrightarrow_R^* x'\}$

Proposition 2.3. Let (Σ, R) be a rewriting system, $(M, \cdot, \lambda)$ a model of R and $f^* : \Sigma^* \to M$ an interpretation. The function $[f^*] : \Sigma^*/R \to M$ defined by $[f^*]([x]) = [f^*(x)](x \in \Sigma^*)$ is a homomorphism. □

Definition 2.21 (rewriting system equivalence)**.** Let (Σ, R_1) and (Σ, R_2) be rewriting systems. R_1 and R_2 are equivalent if and only if Σ^*/R_1 and Σ^*/R_2 are isomorphic.

Finally, we introduce a lemma to compare two rewriting systems that have the same alphabet Σ.

Lemma 2.3. Let (Σ, R_1) and (Σ, R_2) be rewriting systems, and let $(M, \cdot, \lambda)$ be a model of (Σ, R_2). If there exists $(x_1, x_2) \in R_1$ and an interpretation

$f : \Sigma^* \to M$ for Σ^*/R_2 such that $f^*(x_1) \neq f^*(x_2)$, then $\Sigma^*/R_1 \nsim \Sigma^*/R_2$.

Proof. □

3 Quantum circuits

3.1 3 qubits ciruits

3.1.1 Definitions of Quantum Circuits

In this section, we introduce several definitions related to quantum circuits. First, we define quantum bits (qubits), quantum gates, and quantum circuits.

Definition 3.1 (Quantum bits, gates, and circuits)**.** Let $\alpha, \beta \in \mathbb{C}, |0\rangle = (1,0), |1\rangle = (0,1)$ and $m \in \mathbb{N}$.

- A single qubit is denoted by a vector $|x\rangle = \alpha|0\rangle + \beta|1\rangle$.
- A n qubits is denoted by $|x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \in \mathbb{C}^{2^n}$.
- A n qubits quantum gate is an unitary operator $G : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$.
- A quantum circuit Cir of size m is denoted by $Cir = (G_1, G_2, \cdots, G_m)$ where G_i $(i = 1, 2, \cdots, m)$ are n qubits quantum gates.

- An empty circuit is denoted by λ.
- The output of a circuit $Cir = (G_1, G_2, \cdots, G_m)$ for an input $|x\rangle$ is $(G_m \circ \cdots \circ G_2 \circ G_1)|x\rangle$.

Definition 3.2. Let $m, l \in N$, $Cir_1 = (G_1, G_2, \cdots, G_m)$ and $Cir_2 = (G'_1, G'_2, \cdots, G'_l)$ be n qubits quantum circuits. We define an equivalence relation $=_{cir}$ by

$$Cir_1 =_{cir} Cir_2$$
$$\iff \forall |x\rangle \in C^2, (G_m \circ \cdots \circ G_1)|x\rangle = (G'_l \circ \cdots \circ G'_1)|x\rangle.$$

Next, we introduce a quantum gate that plays an important role in proving the universality of quantum circuits.

Definition 3.3. The n qubits controlled-NOT (CNOT) gate is a unitary operator $[c,t]_n : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ $(c, t \in \{1, 2, \cdots, n\})$ defined by

$$\bigotimes_{i=1}^{n} |\delta_i\rangle \mapsto \bigotimes_{i=1}^{t-1} |\delta_i\rangle \otimes |\delta_t \oplus \delta_c\rangle \otimes \bigotimes_{i=t+1}^{n} |\delta_i\rangle.$$

We call c the *control bit* and t the *target bit*. We use a version of Feynmann's notation [Fey85] for diagrammatic representations of CNOT gates (cf. Figure 4). An example of 3 qubits quantum circuits is illustrated in Figure 5. Each gate is applied in turn from left to right to the n qubits.

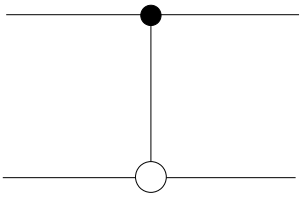

Figure 4: 2 qubits CNOT gate

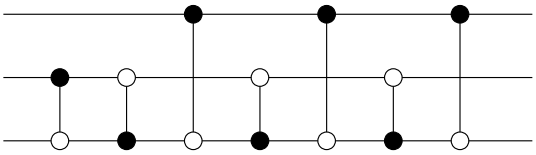

Figure 5: A quantum circuit

Next, we define an equivalence relation between two quantum circuits. This definition is important and allows us to discuss the equivalence of circuits. In this paper, we consider only quantum circuits that are constructed by 3 qubits CNOT gates. We denote as the set of circuits CQC_3 as

$$CQC_3 = \{([c_1, t_1]_3, [c_2, t_2]_3, \cdots, [c_m, t_m]_3)|$$

$$c_i, t_i \in \{1, 2, 3\}, c_i \neq t_i, m \in \mathbb{N}\}.$$

We note that two different circuits Cir_1 and Cir_2 in CQC_3 , may be equiv-

alent in the sense of $=_{cir}$, i.e. $Cir_1 =_{cir} Cir_2$.

Example 3.1. The following equation can be considered as illustrated in Figure.6:

$$([1,2]_3,[2,3]_3) =_{cir} ([2,3]_3,[1,2]_3,[1,3]_3).$$

For all input qubits $|\delta_1\rangle \otimes |\delta_2\rangle \otimes |\delta_3\rangle$, we need to prove the equivalence of the outputs. First, we compute $([1,3]_3 \circ [1,2]_3 \circ [2,3]_3)(|\delta_1\rangle \otimes |\delta_2\rangle \otimes |\delta_3\rangle)$,

$$\begin{aligned} & ([2,3]_3 \circ [1,2]_3)(|\delta_1\rangle \otimes |\delta_2\rangle \otimes |\delta_3\rangle) \\ =_{cir} \quad & [2,3]_3(|\delta_1\rangle \otimes |\delta_1 \oplus \delta_2\rangle \otimes |\delta_3\rangle) \\ =_{cir} \quad & |\delta_1\rangle \otimes |\delta_1 \oplus \delta_2\rangle \otimes |\delta_1 \oplus \delta_2 \oplus \delta_3\rangle. \end{aligned}$$

Next, we compute $([1,3]_3 \circ [1,2]_3 \circ [2,3]_3)(|\delta_1\rangle \otimes |\delta_2\rangle \otimes |\delta_3\rangle)$,

$$\begin{aligned} & ([1,3]_3 \circ [1,2]_3 \circ [2,3]_3)(|\delta_1\rangle \otimes |\delta_2\rangle \otimes |\delta_3\rangle) \\ =_{cir} \quad & ([1,3]_3 \circ [1,2]_3)(|\delta_1\rangle \otimes |\delta_2\rangle \otimes |\delta_2 \oplus \delta_3\rangle) \\ =_{cir} \quad & [1,3]_3(|\delta_1\rangle \otimes |\delta_1 \oplus \delta_2\rangle \otimes |\delta_2 \oplus \delta_3\rangle) \\ =_{cir} \quad & |\delta_1\rangle \otimes |\delta_1 \oplus \delta_2\rangle \otimes |\delta_1 \oplus \delta_2 \oplus \delta_3\rangle. \end{aligned}$$

Thus we have $([1,2]_3,[2,3]_3) =_{cir} ([2,3]_3,[1,2]_3,[1,3]_3)$.

We chose three types of simple equations to construct a string rewriting system.

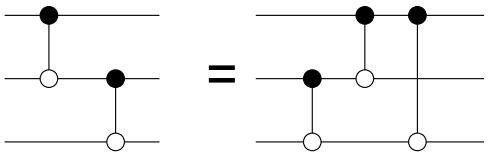

Figure 6: A circuit equation

Definition 3.4. Let G_1, G_2, and $G_3 \in CQC_3$ be CNOT gates.

- For any CNOT gate G, $(G, G) =_{cir} \lambda$ is an eliminated equation.
- $(G_1, G_2) =_{cir} (G_2, G_1)$ is a commutative equation.
- $(G_1, G_2) =_{cir} (G_2, G_1, G_3)$ is an anti-commutative equation .

In this article, we denote six CNOT gates for the 3 qubits $a = [1, 2]_3$, $b = [1, 3]_3$, $c = [2, 1]_3$, $d = [2, 3]_3$, $e = [3, 1]_3$ and $f = [3, 2]_3$.

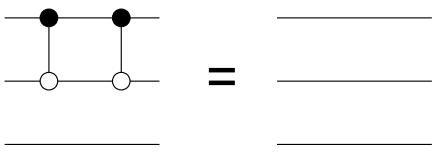

Figure 7: eliminated type: (a , a) = λ

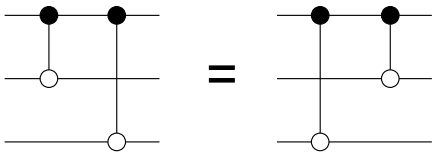

Figure 8: commutative type: (a , b) = (b , a)

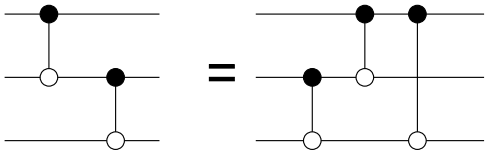

Figure 9: anti-commutative type: $(a, d) =_{cir} (d, a, b)$

3.1.2 Quantum circuit rewriting system

We define a quantum circuit rewriting system for CQC_3.

Definition 3.5 (Quantum circuit rewriting system)**.** Let (Σ, R) be a string rewriting system and $i^* : \Sigma^* \to CQC_3/ =_{cir}$ a function where $\Sigma = \{a, b, c, d, e, f\}$, $i(a) = [1, 2]_3$, $i(b) = [1, 3]_3$, $i(c) = [2, 1]_3$, $i(d) = [2, 3]_3$, $i(e) = [3, 1]_3$ and $i(f) = [3, 2]_3$. (Σ, R) is a quantum circuit rewriting system, if i^* is an interpretation of (Σ, R). We identify a string $w = x_1 x_2 \cdots x_n \in \Sigma^*$ as a circuit

$(i(x_1), i(x_2), \cdots, i(x_n)) \in CQC_3$, and we also call w a *circuit*.

In general, a string rewriting system does not have properties of 'termination' and 'confluence'. So we would like to construct a quantum circuit rewriting system that has both properties termination and confluence. To do so, we use the Knuth-Bendix completion algorithm [KB70, BO93, Met83].

Definition 3.6. Let E be an equation set. If the Knuth-Bendix completion algorithm succeeds for E, then we have a complete transformation rule set R (i.e., a set of transformation rules with the properties of termination and confluence). We denote $KBA(E)$ as the result of the Knuth-Bendix completion algorithm for E.

Example 3.2. Let A be an equation set s.t.

$$A = \left\{ \begin{array}{lll} aa = \lambda, & baba = abab, & dbd = bdb, \\ bb = \lambda, & dbabd = abab, & da = ad, \\ dd = \lambda & & \end{array} \right\}.$$

We can compute $KBA(A)$:

$$KBA(A) = \left\{ \begin{array}{ll} aa \to \lambda, & ababd \to dbab, \\ abadb \to bdba, & abdba \to badb, \\ adbab \to babd, & baba \to abab, \\ babdb \to adba, & badba \to abdb, \\ bb \to \lambda, & bdbab \to abad, \\ da \to ad, & dbabd \to abab, \\ dbad \to bdba, & dbd \to bdb, \\ dd \to \lambda & \end{array} \right\}.$$

Next, we apply the Knuth-Bedix completion algorithm to 18 equations

$$E_{all} = \left\{ \begin{array}{lll} aa = \lambda, & fbfb = a, & ab = ba \\ bb = \lambda, & adad = b, & bd = db \\ cc = \lambda, & dede = c, & cd = dc \\ dd = \lambda, & bcbc = d. & ce = ec \\ ee = \lambda, & fcfc = e, & af = fa \\ ff = \lambda, & eaea = f, & ef = fe \end{array} \right\} \quad (1)$$

introduced by Iwama et. al. 2002 [IKY02]. We note that anti-commutative equations $xy = yxz$ (x, y and $z \in \Sigma$) equivalent to $xyxy = z$ ($xyxy = xyyxz = z$). We also call $xyxy = z$ (x, y and $z \in \Sigma$) anti-commutative equations. We used the *Mathematica* software (version 9) to compute the

complete transformation rule set $KBA(E_{all})$, and we list it in the Appendix. The number of elements of $KBA(E_{all})$ is 114.

$$|KBA(E_{all})| = 114.$$

We note that we have applied an extended Knuth-Bendix completion algorithm which produce an irreducible transformation rule set introduced in [Met83]. The transformation rule set $KBA(E_{all})$ is an irreducible transformation rule set. The number of rules obtained by the original Knuth-Bendix completion algorithm is 244. A string rewriting system $(\Sigma, R_{E_{all}})$ is thus defined where $R_{E_{all}} = KBA(E_{all})$.

We would like to investigate commutativity of E_{all}.

Lemma 3.1. We prove the following equations.

1. $(acac, ca) \in R_{E_{all}}$,
2. $(bebe, eb) \in R_{E_{all}}$ and
3. $(dfdf, fd) \in R_{E_{all}}$.

Proof.

1. First, we show $fbca = caed$. Since $bcbc = d$, $fcfc = e$, $adad = b$ and $eaea = f$, we have $bc = cbd$, $fc = cfe$, $da = bad$ and $ea = fae$. So we

have

$$\begin{aligned} f(bc)a &= f(cbd)a \\ &= (cfe)bda \\ &= cfeb(bad) \\ &= cf(ea)d \\ &= cf(fae)d \\ &= caed. \end{aligned}$$

Since $fbca = caed$ and $fbfb = a$,

$$\begin{aligned} acac &= (fbfb)cac \\ &= fb(caed)c \\ &= (caed)edc \\ &= ca(eded)c \\ &= cacc \\ &= ca. \end{aligned}$$

2. We can prove $bebe = eb$ by the same method to prove $acac = ca$. We rewrite $a \to b$, $b \to d$, $c \to e$, $d \to c$, $e \to f$ and $f \to a$ in the proof of $acac = ca$.

3. We can prove $dfdf = fd$ by the same method to prove $acac = ca$. We rewrite $a \to d$, $b \to c$, $c \to f$, $d \to e$, $e \to a$ and $f \to b$ in the proof of $acac = ca$.

□

Similarly, we obtain the following corollary.

Corollary 3.1.

1. $(caca, ac) \in R_{E_{all}}$,
2. $(ebeb, be) \in R_{E_{all}}$ and
3. $(fdfd, df) \in R_{E_{all}}$.

Proof.

1. Since $acac = ca$ and $cc = \lambda$, we have

$$\begin{aligned} caca &= caca(cc) \\ &= c(acac)c \\ &= c(ca)c \\ &= ac. \end{aligned}$$

2. 3. Similarly, we can prove.

□

By Lemma 3.1 and Corollary 3.1, we have the complete table of $xyxy$ for Σ^*/E_{all}.

x \ y	a	b	c	d	e	f
a	λ	λ	ca	b	f	λ
b	λ	λ	d	λ	eb	a
c	ac	d	λ	λ	λ	e
d	b	λ	λ	λ	c	fd
e	f	be	λ	c	λ	λ
f	λ	a	e	df	λ	λ

Table 2: $xyxy$ for Σ^*/E_{all}

Example 3.3. We show an equation $(ebe, beb) \in \Sigma/R_{E_{all}}$. Since $bb = \lambda$ and $ebeb = be$, we have $ebe = ebe(bb) = (ebeb)b = beb$ and $(ebe, beb) \in \Sigma/R_{E_{all}}$. We note that the rewriting rule $ebe \to beb$ appears on the last 6 line of Appendix.

We prove commutator of the general case. Let $x = [i,j]_3$, $y = [j,i]_3$, $v = [i,k]_3$, $w = [k,j]_3$, $v_y = [j,k]$ and $w_y = [k,i]$ $(i,j,k = 1,2,3,\ i \neq j,\ j \neq k,\ k \neq i)$.

$$x = [i,k]_3[k,j]_3[i,k]_3[k,j]_3 = vwvw.$$

First, we show $vwyx = yxv_yw_y$.

$$wy = yww_y, vy = yvv_y,$$

$$w_yx = wxw_y \text{ and } v_yx = vxv_y.$$

$$\begin{aligned} vwyx &= v(yww_y)x \\ &= (yvv_y)ww_yx \\ &= yvv_yw(wxw_y) \\ &= yvv_yxw_y \\ &= yv(vxv_y)w_y \\ &= yxv_yw_y. \end{aligned}$$

Since $vwyx = yxv_yw_y$, $vwvw = x$ and $v_yw_yv_yw_y = y$.

$$\begin{aligned} xyxy &= (vwvw)yxy \\ &= vw(yxv_yw_y)y \\ &= (yxv_yw_y)v_yw_yy \\ &= yx(v_yw_yv_yw_y)y \\ &= yxyy \\ &= yx. \end{aligned}$$

Proposition 3.1. Let $(\Sigma, R_{E_{all}})$ be a quantum circuit rewriting system where E_{all} a set of equations defined by (1). Then we have followings;

1. $|NF(w)| \leq 6, (w \in \Sigma^7)$,

2. $|NF(w)| \leq 6, (w \in \Sigma^*)$, and

3. $|\Sigma^*/R_{E_{all}}| = 168$.

That is the length of $NF(w)$ is at most 6 for any string $w \in \Sigma^*$ and the number of normal forms is 168.

Proof.

1. We compute the $normalform$ for any string $w \in \Sigma^7$, then we have the length of a $normalform$ is at most 6.

2. For any string $w \in \Sigma^*$ which length is $n \geq 7$, w contain a substring which length is 7. Thus w is rewritten to w' which length is at most $n - 1$. Inductively, for any string $w \in \Sigma^*$, the length of $NF(w)$ is at most 6.

3. We compute the $normalform$ for any string $w \in \Sigma^k$ $(1 \leq k \leq 6)$. So we have all elements of $\Sigma^*/R_{E_{all}}$ and we have $|\Sigma^*/R_{E_{all}}| = 168$.

□

We list the all elements of $\Sigma^*/R_{E_{all}}$ in Appendix. The question now arises: Is the set of equations redundant? Let E_6 be a set of equations such

that

$$E_6 = E_{all} - \left\{ \begin{array}{l} ab = ba \\ bd = db \\ cd = dc \\ ce = ec \\ af = fa \\ ef = fe \end{array} \right\} = \left\{ \begin{array}{ll} aa = \lambda, & fbfb = a \\ bb = \lambda, & adad = b \\ cc = \lambda, & dede = c \\ dd = \lambda, & bcbc = d \\ ee = \lambda, & fcfc = e \\ ff = \lambda, & eaea = f \end{array} \right\}. \tag{2}$$

The size of this equation set is $|E_6| = 12$.

Lemma 3.2. We prove the following equations.

1. $(ba, ab) \in R_{E_6}$,
2. $(db, bd) \in R_{E_6}$,
3. $(dc, cd) \in R_{E_6}$,
4. $(ec, ce) \in R_{E_6}$,
5. $(fa, af) \in R_{E_6}$, and
6. $(fe, ef) \in R_{E_6}$.

Proof.

1. Since $adad = b$ and $aa = bb = dd = \lambda$, we have $ba = ba(bb) = (adad)a(adad)b = ab$ and $(ba, ab) \in R_{E_6}$.

2. 3. 4. 5. 6. We can prove similarly.

□

We compute a complete transformation rule set $KBA(E_6)$ by using the Knuth-Bendix completion algorithm, and we can have $KBA(E_6) = KBA(E_{all})$. The above results mean that commutative type equations is not required for the initial equation set. We have the next proposition.

Proposition 3.2. Let (Σ, R) be a quantum circuit rewriting system, E_{all} and E_6 sets of equations defined by (1) and (2),

$$\Sigma^*/R_{E_6} = \Sigma^*/R_{E_{all}}.$$

□

In the following section, we reduce the size of an equation set and show the existence of the minimal set of equations E_{min} of E_6 that generates the isomorphic monoid $\Sigma^*/R_{E_{min}} = \Sigma^*/R_{E_6}$.

3.1.3 Minimal set of equations

Definition 3.7 (Minimal set of equations)**.** Let $E \subseteq \Sigma^* \times \Sigma^*$. A subset $E_{min} \subset E$ is a minimal equation set of E if and only if

- $\Sigma^*/R_{E_{min}} = \Sigma^*/R_E$, and

- If $\Sigma^*/R_{E'} = \Sigma^*/R_E$ then $|E_{min}| \leq |E'|$ for all $E' \subset E$.

In this section, we investigate a minimal set of equations of E_6 such that $\Sigma^*/R_{E_{min}} = \Sigma^*/R_{E_6}$. We delete some equations from E_6 and prove that the factor monoids of the equations are isomorphic. We follow the same line of thought as was used for the elementary Tietze transformation [BO93]. We first prove the following proposition.

Proposition 3.3. Let (Σ, R) be a quantum circuit rewriting system, E_6 a set of equations defined by (2),

$$E_5 = \left\{ \begin{array}{ll} aa = \lambda, & fbfb = a \\ bb = \lambda, & adad = b \\ cc = \lambda, & dede = c \\ dd = \lambda, & bcbc = d \\ ee = \lambda, & fcfc = e \\ & eaea = f \end{array} \right\}, \text{and}$$

$$E_2 = \left\{ \begin{array}{ll} aa = \lambda, & fbfb = a \\ bb = \lambda, & adad = b \\ & dede = c \\ & bcbc = d \\ & fcfc = e \\ & eaea = f \end{array} \right\}. \tag{3}$$

Then we have followings:

1. $(efc, cf), ((fc)e, e(fc)) \in R_{E_5}$,
2. $(ff, \lambda) \in R_{E_5}$,
3. $\Sigma^*/R_{E_5} = \Sigma^*/R_{E_6}$, and
4. $\Sigma^*/R_{E_2} = \Sigma^*/R_{E_6}$.

Proof. We prove this proposition in following procedures.

1. Since $aa = ee = cc = \lambda$, $eaea = f$ and $fcfc = e$, we have

$$\begin{aligned} cf &= (ee)(aeaeeaea)cf(cc) \\ &= e(eaea)e(eaea)cfcc \\ &= efe(fcfc)c \\ &= efeec \\ &= efc \end{aligned}$$

and $(efc, cf) \in R_{E_5}$.

Since $fcfc = e$, we have

$$\begin{aligned} fce &= fc(fcfc) \\ &= efc, \end{aligned}$$

and $((fc)e, e(fc)) \in R_{E_5}$.

2. Since $cc = \lambda$, $efc = cf$, $fce = efc$ and $fcfc = e$, we have

$$\begin{aligned} ff &= f(cc)f \\ &= fc(efc) \\ &= (efc)fc \\ &= e(e) \\ &= \lambda, \end{aligned}$$

and $(ff, \lambda) \in R_{E_5}$.

3. We show that $\Sigma^*/R_{E_5} = \Sigma^*/R_{E_6}$. Since $[ff]_{E_5} = [\lambda]_{E_5}$, we have $\Sigma^*/R_{E_5} = \Sigma^*/R_{E_6}$.

4. Let E_4, E_3 and E_2 be sets of equations where

$$\begin{aligned} E_4 &= E_6 - \{ee = \lambda, ff = \lambda\}, \\ E_3 &= E_6 - \{dd = \lambda, ee = \lambda, ff = \lambda\}, and \\ E_2 &= E_6 - \{cc = \lambda, dd = \lambda, ee = \lambda, ff = \lambda\}. \end{aligned}$$

Similarly, we have $(ee, \lambda) \in R_{E_4}$, $(dd, \lambda) \in R_{E_3}$ and $(cc, \lambda) \in R_{E_2}$. So we obtain

$$\begin{aligned} \Sigma^*/R_{E_6} &= \Sigma^*/R_{E_5} = \Sigma^*/R_{E_3} = \Sigma^*/R_{E_4} \\ &= \Sigma^*/R_{E_2}. \end{aligned}$$

□

Next, we prove that E_2 is a minimal set of equations.

Proposition 3.4.

Let $E' \subset E_6$. If $\Sigma^*/R_{E'} = \Sigma^*/R_{E_6}$, then $|E'| \geq 8$.

Proof. We will prove this in two steps.

step 1: First, we prove that we cannot remove an anti-commutative equation from E_6 .

We define a set of equations

$$E_{anti} = \left\{ \begin{array}{l} fbfb = a, adad = b, dede = c, \\ bcbc = d, fcfc = e, eaea = f \end{array} \right\}.$$

Let $u \in E_{anti}$ and consider u as $xyxy = z$ $(x, y, z \in \Sigma)$. We define E_u as $E_u = E_6 - \{u\} = E_6 - \{xyxy = z\}$. Then we can show $\Sigma^*/R_{E_u} \neq \Sigma^*/R_{E_6}$ as follows. We consider a monoid $M = (\{0, 1\}, \cdot, 0)$ where a

binary operator $\cdot \subset \{0,1\} \times \{0,1\} \to \{0,1\}$ is defined by Table 3, and a function $i : \Sigma^* \to M$ is defined as

$$i(\lambda) = i(k) = 0(\forall k \neq z), \text{ and } \ i(z) = 1.$$

We consider a homomorphism i^* and show that i^* is an interpretation for E_u:

$$i^*(kk) = 0 \cdot 0 = 0 = i^*(\lambda),$$

$$i^*(zz) = 1 \cdot 1 = 0 = i^*(\lambda), \text{ and}$$

$$\begin{aligned} i^*(mnmn) &= i^*(mn) \cdot i^*(mn) \\ &= 0 \\ &= i^*(k), \forall m, n \in \Sigma, k \neq z. \end{aligned}$$

Since $x \neq z$ and $y \neq z$, then the value of $i^*(xyxy)$ is

$$i^*(xyxy) = i(0) \cdot i(0) \cdot i(0) \cdot i(0) = 0 \cdot 0 \cdot 0 \cdot 0 = 0.$$

Since $i^*(z)$ is

$$i^*(z) = i(z) = 1.$$

We have $i^*(xyxy) \neq i^*(z)$. By Lemma 2.3, $[xyxy]_{E_u} \neq [z]_{E_u}$. On the other hand, it is obvious that $[xyxy]_{E_6} = [z]_{E_6}$. Therefore,

$$\Sigma^*/R_{E_u} \neq \Sigma^*/R_{E_6}.$$

$\cdot$	0	1
0	0	1
1	1	0

Table 3: Definition of the binary operator $\cdot$

step 2: Let $x \in \Sigma$ and E_x a set of equations defined by

$$E_x = \left\{ \begin{array}{ll} xx = \lambda, & fbfb = a \\ & adad = b \\ & dede = c \\ & bcbc = d \\ & fcfc = e \\ & eaea = f \end{array} \right\}.$$

Then we can have $\Sigma^*/R_{E_x} \neq \Sigma^*/R_{E_6}$ as follows. For example, if we consider $x = a$, we can prove $\Sigma^*/R_{E_a} \neq \Sigma^*/R_{E_6}$. Let $N = (\{0,1,2\}, \cdot, 0)$ be a monoid where a binary operator $\cdot \subset \{0,1\} \times \{0,1\} \to \{0,1\}$ is defined by Table 4, and let the function $j : \Sigma^* \to N$ be $j(\lambda) = j(a) = j(c) = 0, j(b) = j(e) = 1$ and $j(d) = j(f) = 2$. We consider a homomorphism j^* and show that j^* is an interpretation for

E_a:

$$j^*(aa) = j(a) \cdot j(a) = 0 = j^*(\lambda),$$

$$j^*(fbfb) = j^*(fb) \cdot j^*(fb) = 0 \cdot 0 = 0 = j^*(a),$$

$$j^*(adad) = j^*(ad) \cdot j^*(ad) = 2 \cdot 2 = 1 = j^*(b),$$

$$j^*(dede) = j^*(de) \cdot j^*(de) = 0 \cdot 0 = 0 = j^*(c),$$

$$j^*(bcbc) = j^*(bc) \cdot j^*(bc) = 1 \cdot 1 = 2 = j^*(d),$$

$$j^*(fcfc) = j^*(fc) \cdot j^*(fc) = 2 \cdot 2 = 1 = j^*(e) \text{and}$$

$$j^*(eaea) = j^*(ea) \cdot j^*(ea) = 1 \cdot 1 = 2 = j^*(d).$$

The value of $j^*(bb)$ is

$$j^*(bb) = j(b) \cdot j(b) = 1 \cdot 1 = 2.$$

The value of $j^*(\lambda)$ is

$$j^*(\lambda) = j(\lambda) = 0.$$

Since $j^*(bb) \neq j^*(\lambda)$, we have $[bb]_{E_a} \neq [\lambda]_{E_a}$ by Lemma 2.3 . On the other hand, it is obvious that $[bb]_{E_6} = [\lambda]_{E_6}$. Therefore,

$$\Sigma^*/R_{E_a} \neq \Sigma^*/R_{E_6}.$$

$\cdot$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 4: Definition of the binary operator $\cdot$

Let $E' \subset E_6$ be a set of equations. If $\Sigma^*/R_{E'} = \Sigma^*/R_{E_6}$, it contained at least six anti-commutative equations by step1 and at least two eliminated equations by step2. Therefore, if $\forall E' \subset E_6$ it holds that $\Sigma^*/R_{E'} = \Sigma^*/R_{E_6}$, then

$$|E'| \geq 8.$$

□

Lemma 3.3. Let E_{ac} be a set of equations where

$$E_{ac} = \left\{ \begin{array}{ll} aa = \lambda, & fbfb = a \\ & adad = b \\ cc = \lambda, & dede = c \\ & bcbc = d \\ & fcfc = e \\ & eaea = f \end{array} \right\}.$$

Then

$$\Sigma^*/R_{E_{ac}} \neq \Sigma^*/R_{E_6}. \tag{4}$$

Proof. We show $[bb]_{E_{ac}} \neq [\lambda]_{E_{ac}}$. Let $N = (\{0,1,2\}, \cdot, 0)$ be a monoid where a binary operator $\cdot \subset \{0,1\} \times \{0,1\} \to \{0,1\}$ is defined by Table 4, and let the function $j : \Sigma^* \to N$ be $j(\lambda) = j(a) = j(c) = 0, j(b) = j(e) = 1$ and $j(d) = j(f) = 2$. The function j is the same function used in step 2 of Proposition 3.4. We consider a homomorphism j^* and show that j^* is an interpretation for E_a. We check that $j^*(cc) = j^*(\lambda)$.

$$j^*(cc) = j(c) \cdot j(c) = 0 = j^*(\lambda).$$

The value of $j^*(bb)$ is

$$j^*(bb) = j(b) \cdot j(b) = 1 \cdot 1 = 2.$$

The value of $j^*(\lambda)$ is

$$j^*(\lambda) = j(\lambda) = 0.$$

Since $j^*(bb) \neq j^*(\lambda)$, we have $[bb]_{E_{ac}} \neq [\lambda]_{E_{ac}}$ by Lemma 2.3. On the other hand, it is obvious that $[bb]_{E_6} = [\lambda]_{E_6}$. Therefore,

$$\Sigma^*/R_{E_{ac}} \neq \Sigma^*/R_{E_6}.$$

□

From the above discussion, we derive the next theorem.

Theorem 3.1. There exists a minimal equation set E_{min} of E_6 such that $|E_{min}| = 8$.

Proof. E_2 is a minimal equation of E_6 by Proposition 3.3 and Proposition 3.4. □

Next, we show that there is an 8 element set of equations $F_2 \nsubseteq E_6$ such that $\Sigma^*/R_{F_2} = \Sigma^*/R_{E_6}$.

Proposition 3.5. Let $F_2 \nsubseteq E_6$ be a set of equations defined by

$$F_2 = \left\{ \begin{array}{ll} aa = \lambda, & bfbf = a \\ bb = \lambda, & dada = b \\ & eded = c \\ & cbcb = d \\ & cfcf = e \\ & aeae = f \end{array} \right\}.$$

Then $\Sigma^*/R_{F_2} = \Sigma^*/R_{E_6}$.

Proof. Let F_{anti} and F_6 be sets of equations defined by

$$F_{anti} = \left\{ \begin{array}{lll} bfbf = a, & dada = b, & eded = c \\ cbcb = d, & cfcf = e, & aeae = f \end{array} \right\}, \text{ and}$$

$$F_6 = \left\{ \begin{array}{ll} aa = \lambda, & bfbf = a \\ bb = \lambda, & dada = b \\ cc = \lambda, & eded = c \\ dd = \lambda, & cbcb = d \\ ee = \lambda, & cfcf = e \\ ff = \lambda, & aeae = f \end{array} \right\}.$$

We can prove

$$\Sigma^*/R_{F_2} = \Sigma^*/R_{F_6} \tag{5}$$

by following the same method in Proposition 4. Since we can prove $bfbf = a, dada = b, eded = c, cbcb = d, cfcf = e$ and $aeae = f$ in F_6, we have

$$\Sigma^*/R_{F_6} = \Sigma^*/R_{E_6 \cup F_{anti}}. \tag{6}$$

Similarly, we can have

$$\Sigma^*/R_{E_6 \cup F_{anti}} = \Sigma^*/R_{E_6}. \tag{7}$$

By (5), (6), and (7), we have $\Sigma^*/R_{F_2} = \Sigma^*/R_{E_6}$. □

3.2 General Case

In this section, we consider how to extend quantum circuits rewriting system to higher qubits circuits. There exists two CNOT gates $[1, 2]$ and $[2, 1]$ in 2 qubits rewriting system. We should define commutators as following in the

Table 5. In 3 qubits rewriting system, we add 4 new gates $[1,3]$, $[2,3]$, $[3,1]$

x \ y	$[1,2]$	$[2,1]$
$[1,2]$	λ	$[2,1][1,2]$
$[2,1]$	$[1,2][2,1]$	λ

Table 5: 2 qubits commutator $xyxy$

and $[3,2]$. We should suppose commutators between $[2,3]$ and $[3,2]$ such that we need to define the commutators $\{[1,2],[2,1],[1,3],[2,3],[3,1],[3,2]\}$.

x \ y	$[2,3]$	$[3,2]$
$[2,3]$	λ	$[3,2][2,3]$
$[3,2]$	$[2,3][3,2]$	λ

Table 6: 2 qubits commutator $xyxy$

Since proposition 3.4 step 2, it is necessary to suppose 6 anti-commutative type equations that decompose gates $[i,j]$ by two gates,

$$\exists k \neq i \; and \; k \neq j, [i,j] = [i,k][k,j][i,k][k,j].$$

Equations for a general case can be defined by equations of commutators.

We extend our formulation for n qubits CNOT gates. First, we calculate the number of CNOT gates for n qubits. For example, there are 6 CNOT

x \ y	$[1,2]$	$[2,1]$	$[2,3]$	$[3,2]$	$[1,3]$	$[3,1]$
$[1,2]$	λ	$[2,1][1,2]$	$[1,3]$	λ		$*_{1,6} = [3,2]$
$[2,1]$	$[1,2][2,1]$	λ	λ	$[3,1]$	$*_{2,5} = [2,3]$	
$[2,3]$	$[1,3]$	λ	λ	$[3,2][2,3]$		$*_{3,6} = [2,1]$
$[3,2]$	λ	$[3,1]$	$[2,3][3,2]$	λ	$*_{4,5} = [1,2]$	
$[1,3]$						
$[3,1]$						

Table 7: 3 qubits $xyxy$

gates for 3 qubits and 12 gates for 4 qubits. Let Σ_n be an alphabet of n qubits quantum rewriting system such that $\Sigma_n = \{[i,j]_n | i,j = 1,2,\cdots n\}$. We count the permutation of 2 qubits,

$$|\Sigma_n| = n(n-1).$$

Hence, the number of CNOT gates for n qubits is $n(n-1)$.

Next, we consider the number of equations of CNOT circuits for n qubits. For example, there are 18 circuit equations for 3 qubits and 72 circuit equations for 4 qubits. For n qubits, For all $i,\ j = 1,2,\cdots,n$, there exists a eliminate type equation $[i,j]_n[i,j]_n = \lambda$. Thus, the number of eliminate

type equations is

$$n(n-1).$$

Let $[i,j]_n$ and $[i',j']_n$ be n qubits CNOT gates. If $i \neq j'$ and $j \neq i'$, then there exists a commutative equation ($[i,j]_n[i',j']_n = [i',j']_n[i,j]_n$). Thus , the number of commutative type equations is

$$\frac{n(n-1)^2(n-2)}{2}.$$

If ($i = j'$ and $j \neq i'$) or ($j = i'$ and $i \neq j'$), then there exists a anti-commutative type of equation. Thus, the number of anti-commutative type of equations is

$$n(n-1)(n-2).$$

Let Eq_n be a set of equations for n qubits. The number of equations of CNOT circuits for n qubits

$$\begin{aligned} |Eq_n| &= n(n-1) + \frac{n(n-1)^2(n-2)}{2} + n(n-1)(n-2) \\ &= \frac{(n(n-1))^2}{2}. \end{aligned}$$

Therefore, we are able to discuss about n qubits quantum rewriting systems.

1. If $i_1 = i_2$ and $j_1 = j_2$, then $[i_1,j_1]_n[i_1,j_1]_n = \lambda$.

2. If $i_1 = i_2$ and $j_1 \neq j_2$, then $[i_1,j_1]_n[i_1,j_2]_n = [i_1,j_2]_n[i_1,j_1]_n$.

3. If $i_1 \neq i_2$ and $j_1 = j_2$, then $[i_1,j_1]_n[i_2,j_1]_n = [i_2,j_1]_n[i_1,j_1]_n$.

4. If $i_1 \neq i_2$, $j_1 \neq j_2$, $i_1 \neq j_2$ and $j_1 \neq i_2$, then $[i_1, j_1]_n[i_2, j_2]_n = [i_2, j_2]_n[i_1, j_1]_n$.

5. If $i_1 = j_2$ and $j_1 = i_2$, then $[i_1, j_1]_n[j_1, i_1]_n = [j_1, i_1]_n[i_1, j_1]_n[j_1, i_1]_n[i_1, j_1]_n$.

6. If $i_1 = j_2$ and $j_1 \neq i_2$, then $[i_1, j_1]_n[i_2, i_1]_n = [i_2, i_1]_n[i_1, j_1]_n[i_2, j_1]_n$.

7. If $i_1 \neq j_2$ and $j_1 = i_2$, then $[i_1, j_1]_n[j_1, j_2]_n = [j_1, j_2]_n[i_1, j_1]_n[i_1, j_2]_n$

4 Implementations using Mathematica

We introduce Mathematica functions to investigate string rewriting systems. We used the Mathematica software (version 9) to compute complete transformation rule sets.

4.1 The Knuth-Bendix completion algorithm

Let R be a set of rewriting rules, $\succ$ an order of strings and CP a set of critical pairs. We introduce the Knuth-Bendix completion algorithm.

1. Input : CP, $\succ$
2. While $CP \neq \pi$
3. begin
4. $(p, q) \in CP$
5. $CP = CP - (p, q)$
6. Compute normal forms of p and q for R
7. if $p = q$
8. then begin
9. $(\alpha, \beta) =$ `analyze`$(NF(p), NF(q) \succ)$
10. $R = R \cup \{(\alpha, \beta)\}$
11. $CP = CP \bigcup_{r \in R}$(set of critical pairs of (α, β) and r)
12. end
13. end
14. return R

The `analyze` procedure outputs a new rule observing the order $\succ$ or error.

1. Input : s, t, $\succ$

2. if $s \succ t$

3. then $(\alpha, \beta) = (s, t)$

4. else if $t \succ s$

5. then $(\alpha, \beta) = (t, s)$

6. else Error

7. return (α, β)

4.2 `Critical`, `MakeCriticalPair`

The source code of components used in the Knuth-Bendix completion algorithm is listed in this section. The function '`IterationLimit`' gives the maximum length of evaluation chain used in trying to evaluate any expression. When we compute normal forms for a rewriting system, it is necessary to repeat many evaluations.

```
$IterationLimit = 10^15;
```

In the Knuth-Bendix completion algorithm, it is necessary to compute critical pairs from an equation set. We compute critical pairs using the following

functions. The function `Critical[x]` has an argument $x = ((s_1, t_1), (s_2, t_2))$ that means a pair of transformation rules.

```
(*x = (s1->t1, s2->t2)*)
Critical[x_] :=
 If[
  x[[1]] == x[[2]]
  (*Same transformation rule does not make critical pair*),
  {},
{StringReplacePart[x[[2, 1]], x[[1, 2]], #],
   StringReplace[StringReplacePart[x[[2, 1]], x[[1, 1]], #],
    x[[2, 1]] -> x[[2, 2]]]} & /@
 Flatten[{StringPosition[x[[2, 1]], x[[1, 1]]]}~
   Join~(StringPosition[x[[2, 1]], StartOfString ~~ #] & /@
     Rest[StringCases[x[[1, 1]], __, Overlaps -> True]])
   ~Join~(StringPosition[x[[2, 1]], # ~~ EndOfString]
   & /@ (Rest[
           StringCases[
         StringReverse[x[[1, 1]]], __, Overlaps -> True]]
         //StringReverse))
         , 1]
  ]
```

Example 4.1. We find overlap points from each starting point, and rewrite

the overlap string by each transformation rules. We have computed critical pairs for a pair of $xaxa \to ab$ for $axa \to c$.

Let X and Y be variables. For $r_1 : s_1 \to t_1 = xaxaX \to abX$, $r_2 : s_2 \to t_2 = axaY \to cY$, r_2 overlap with r_1 at 2 (case 1)and $2 \cdot 2$ (case 2) and r_1 overlap with r_2 at 2 (case 3).

case 1 $\theta = \{X \to Y\}$

$$\theta(s_1[2 \leftarrow t_2]) = xcY,\ \theta(t_1) = abY,$$

case 2 $\theta = \{X \to xaY\}$

$$\theta(s_1[2 \cdot 2 \leftarrow t_2]) = xaxcY,\ \theta(t_1) = abxaY,$$

case 3 $\theta = \{Y \to xaX\}$

$$\theta(s_2[2 \cdot 2 \leftarrow t_1]) = aabX,\ \theta(t_2) = cxaX.$$

```
In[573]:= Critical[{{"axa", "c"}, {"xaxa", "ab"}}]


Out[573]= {{"xc", "ab"}, {"cxa", "aab"}, {"xaxc", "abxa"}}
```

We identify a pair of strings $x = (s, t)$ as a transformation rule $s \to t$. The function `Rule[`x`]` is a type conversion function from a pair of strings to a transformation rule.

```
Rule[x_] :=  If[x == {}, {}, {x[[1]] -> x[[2]]}]
```

```
In[578]:= Rule[{"axa", "c"}]


Out[578]= {"axa" -> "c"}
```

The function `Makerules` transform from a list of pairs to a list of transformation rules.

```
(*A list of pairs to a list of transformation rules*)

Makerules[list_] := Flatten[Rule /@ list]
```

```
In[575]:=

Makerules[{{"axa", "c"}, {"xaxa", "ab"}}]


Out[575]= {"axa" -> "c", "xaxa" -> "ab"}
```

We compute all critical pairs to construct a complete transformation rule set.

```
(*A list of pairs to a list of critical pairs*)

MakeCriticalPair[list_]

        := Flatten[Critical /@ Tuples[{list, list}], 1]
```

```
(*A list of pairs and a pair to a list of critical pairs*)

MakeNewCriticalPair[list_, rule_] :=

 Flatten[Critical /@ (Tuples[{list, rule}]~Union~

     Tuples[{rule, list}]), 1]
```

Example 4.2. We compute a set of critical pairs for the set of transformation rules $axa \to c, xaxa \to ab$.

```
In[576]:= MakeCriticalPair[{{"axa", "c"}, {"xaxa", "ab"}}]



Out[576]= {{"xc", "ab"}, {"cxa", "aab"}, {"xaxc", "abxa"},

{"ab",  "xc"}, {"abxa", "xcxa"}, {"aab", "cxa"}}
```

4.3 `Order`, `Normlform`

An order of string is important to succeed the Kunth-Bendix completion algorithm. We used the following order.

1. The one which the length of string is smaller is smaller.

2. If two strings have the same length, we apply the lexicographical order for the strings.

The function `Order[`(s,t)`]` corrects the order.

```
(*A order of strings*)

Order[{x_, y_}] :=

 If[StringLength[x] > StringLength[y], {{x, y}},

  If[StringLength[y] > StringLength[x], {{y, x}},

   If[x == y, {},

    If[Sort[{x, y}] === {x, y}, {{y, x}},

     {{x, y}}

     ]]

   ]]
```

Example 4.3. We have checked orders of several pairs of strings.

```
In[579]:= Order[{"bac", "ab"}]


Out[579]= {{"bac", "ab"}}


In[580]:= Order[{"ab", "bac"}]


Out[580]= {{"bac", "ab"}}
```

```
In[581]:= Order[{"abc", "bac"}]


Out[581]= {{"bac", "abc"}}
```

In the Kuntuh-Bendix completion algorithm, it is necessary to compute normal forms of each element of critical pairs.

```
(*Computeing a normal form*)

Normalform[pair_, list_] :=

 FixedPoint[StringReplace[#, Makerules[list]] &, #] & /@ pair
```

Example 4.4. Let `pair` be a pair of strings and `list` a set of transformation rules. The function `Normalform[pari, list]` compute a normal form of `pair` in `list`.

```
 In[31]:= Normalform[{"abababcb",

  "cacbac"}, {{"aa", ""}, {"ababc", "caba"}, {"abaca",

   "bcab"}, {"abcab", "baca"}, {"acaba", "babc"}, {"baba",

   "abab"}, {"babca", "acab"}, {"bacab", "abca"}, {"bb",

   ""}, {"bcaba", "abac"}, {"cabab", "abac"}, {"cabac",

   "abab"}, {"cabc", "acab"}, {"cac", "aca"}, {"cb", "bc"}, {"cc",

   ""}}]


Out[31]= {"babc", "bab"}
```

4.4 KnuthBendix

In the Knuth-Bendix completion algorithm, the first thing we have to do is to compute critical pairs for an initial equation set. We construct new transform rules from a set of critical pairs, and reduce elements of the set. We repeat the procedure until the set of critical pairs becomes empty.

```
 (*A Knuth-Bendix completion algorithm*)
KnuthBendix[list_] :=
 Module[{CP, f},
  CP = MakeCriticalPair[list];
  f[cp_, l_] :=
   If[
    cp === {}, l,
    If[Order[Normalform[cp[[1]], l]] === {}, f[Rest[cp], l],
     f[(Rest[cp]~Join~
        MakeNewCriticalPair[l, Order[Normalform[cp[[1]], l]]]),
      l~Join~Order[Normalform[cp[[1]], l]]]
    ];
  f[CP, list]
  ]
```

Example 4.5. Let `example` be a set of string pairs.

```
example =
```

```
{{"aa", ""}, {"bb", ""}, {"cc", ""}, {"baba", "abab"},
{"cb", "bc"}, {"cac", "aca"}, {"cbaba", "abac"}};
```

For the set '`example`', we are able to compute a complete transformation rule set. The computation time of `KnuthBendix[example]` take 0.36 second (Core 2 Duo 2.13 GHz, 4GB).

```
In[25]:= KnuthBendix[example] // Union


Out[25]= {{"aa", ""}, {"ababc", "caba"}, {"abaca", "bcab"},
 {"abcab", "baca"}, {"acaba", "babc"}, {"baba", "abab"},
 {"babca", "acab"}, {"bacab", "abca"}, {"bb", ""},
 {"bcaba", "abac"}, {"cabab", "abac"}, {"cabac", "abab"},
 {"cabc", "acab"}, {"cac",  "aca"}, {"cb", "bc"},
 {"cbaba", "abac"}, {"cc", ""}}
```

4.5 `Irreduce`

In generally, the result of the Knuth-Bendix completion algorithm is not irreducible. Therefore we apply the function `Irreduce[x]` to transform the set to be irreducible.

```
(*A transformation rule set to an irreducible set*)
Irreduce[list_] := Module[{comp},
```

```
comp[l_] :=
 Module[{step1, list2, step2},
  step1[pair_] := {pair[[1]],
    FixedPoint[StringReplace[pair[[2]], Makerules[list]] &,
     pair[[2]]]};
  list2 = step1 /@ l;
  step2[
    pair_] := {FixedPoint[
     StringReplace[pair[[1]],
       Makerules[Cases[list2, Except[pair]]]] &, pair[[1]]],
    pair[[2]]};
  DeleteCases[Union[(step2 /@ list2)], {x__, x__}]
  ];
FixedPoint[comp, list]
]
```

Example 4.6. A small example of the Knuth-Bendix completion algorithm.

We can reduce the element {cbaba, abac} of the result of `KnuthBendix[example]`.

Since $cb \to bc$ and $bcaba \to abac$, we have

$$cbaba \to bcaba \to abac,$$

and the result is reducible. Therefore we are able to improve the set `KnuthBendix[example]`.

```
In[24]:= KnuthBendix[example] // Irreduce


Out[24]= {{"aa", ""}, {"ababc", "caba"}, {"abaca", "bcab"},

{"abcab",  "baca"}, {"acaba", "babc"}, {"baba", "abab"},

{"babca", "acab"}, {"bacab", "abca"}, {"bb", ""},

{"bcaba", "abac"}, {"cabab", "abac"}, {"cabac", "abab"},

{"cabc", "acab"}, {"cac", "aca"}, {"cb", "bc"}, {"cc", ""}}
```

4.6 Cayley graph

We used equations that are introduced by Iwama et. al. to compute complete transformation rule sets. Since there may be other efficient equations, we considered Cayley graphs to find other possible equations. Cayley graphs for quantum circuit rewriting systems could be a good hint to find them. We were able to construct a Cayley graph for a 3 qubits quantum circuit rewriting system $(\{a, b, c, d, e, f\}, R)$. The graph shows how circuits are transformed by CNOT gates. The diameter D_{abcdef} is 6.

$$D_{abcdef} = 6.$$

The result suppose that the length of normal form is at most 6 for 3 qubits quantum circuit rewriting system $(\{a, b, c, d, e, f\}, R)$ in Figure 10. Next we

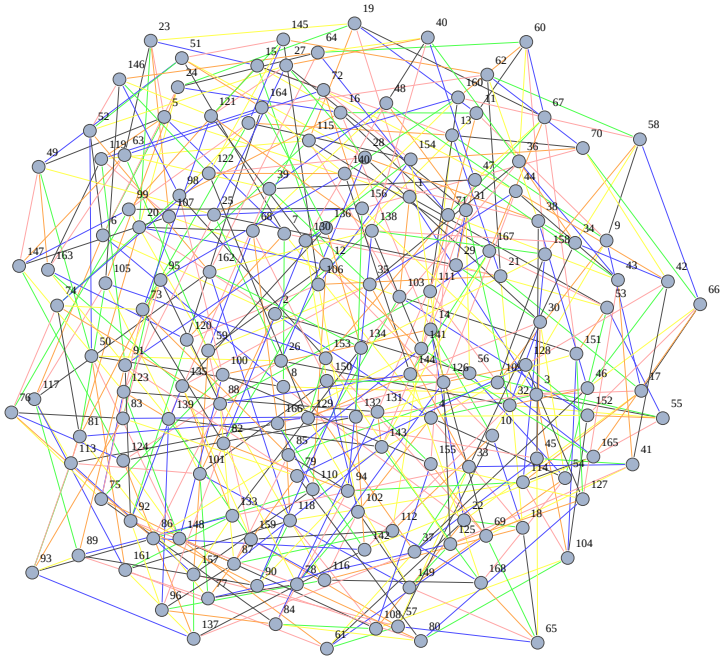

Figure 10: The Cayley graph for $\{a, b, c, d, e, f\}$

consider another 3 qubits quantum circuit rewriting system $(\{a, c, d, f\}, R)$ that uses a different alphabet set. Since $b = adad$ and $e = cfcf$, the rewriting system $(\{a, c, d, f\}, R)$ and $(\{a, b, c, d, e, f\}, R)$ are equivalent. We calculate the complete transformation rule set for $(\{a, c, d, f\}, R)$ in Figure 11.

```
Eshort = {{"aa", ""}, {"cc", ""}, {"dd", ""}, {"ff", ""},
{"eaea", "f"}, {"dc", "cd"}, {"fa", "af"}, {"cac", "aca"},
{"fdf", "dfd"}, {"dada", "adad"}, {"fcfc", "cfcf"}};
```

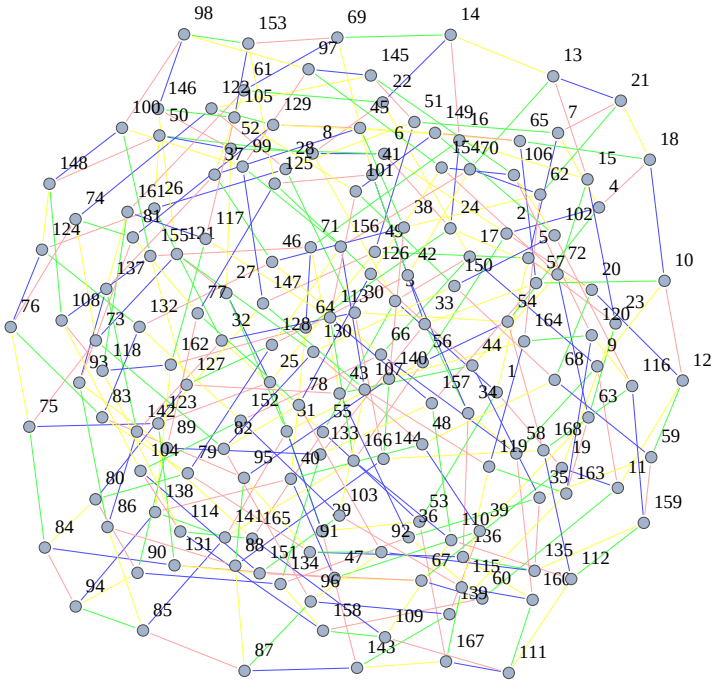

Figure 11: The Cayley graph for $\{a, c, d, f\}$

We calculate the Cayley graph for $(\{a, c, d, f\}, R)$. The diameter D_{acdf} is 8.

$$D_{acdf} = 8.$$

The maximum length of normal forms for 3 qubits quantum circuit rewriting systems are not always same. If we add new quantum gate that are easy to create, we will construct normal forms that are smaller than previous one.

As a result of our computing of the Cayley graphs we are able to check transformations that CNOT gates operate each circuits. It is difficult to

find minimum circuits in the graph. Therefore we computed the shortest paths of the graph between λ and each vertexes. We show the result of shortest paths graph for $(\{a, b, c, d, e, f\}, R)$ and $(\{a, c, d, f\}, R)$ in Figure 12 and Figure 13, respectively. We are able to check 168 minimum circuits for each vertexes. The figure shows an example of minimum circuits. A path from center vertex to a leaf vertex corresponds to a circuits. We note that a path does not always correspond to an our normal form.

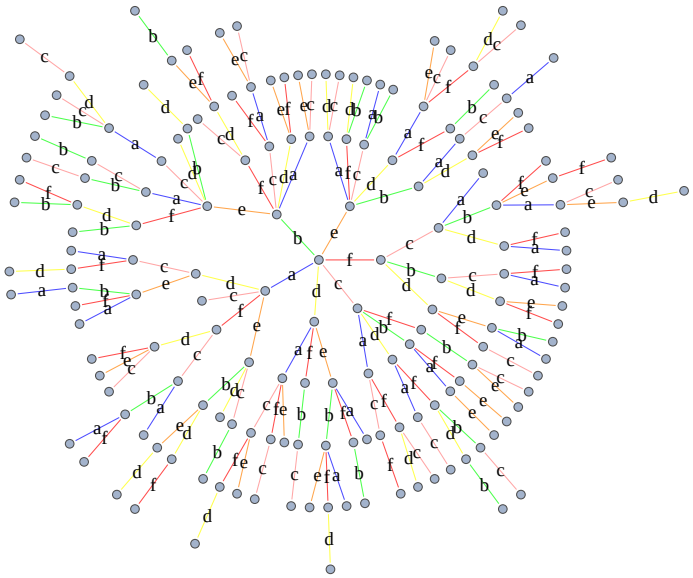

Figure 12: The graph of shortest path from λ for $(\{a, b, c, d, e, f\}, R)$

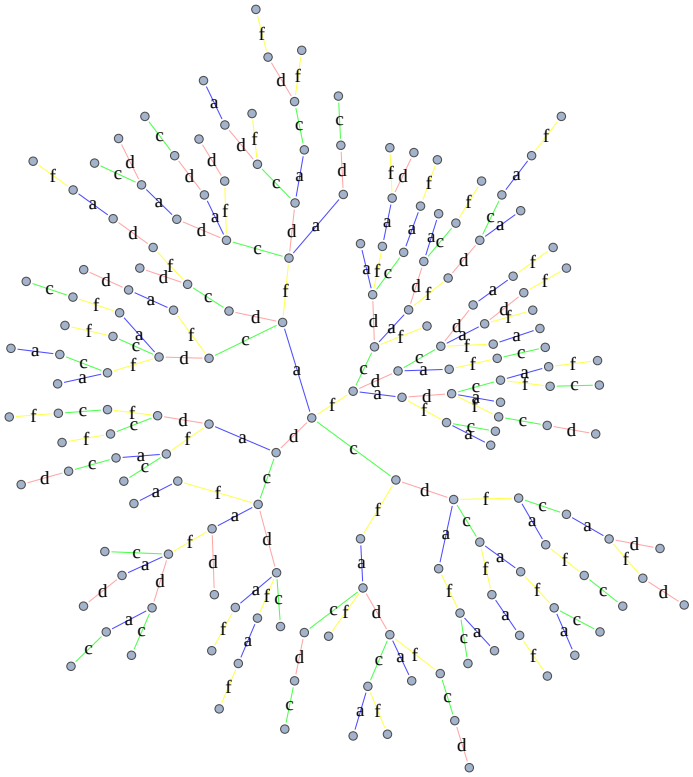

Figure 13: The graph of shortest path from λ for $(\{a, c, d, f\}, R)$

5 Conclusion & Future Work

We considered rewriting systems in order to reduce the size of quantum circuits. We compute a set of complete transformation rules using the Knuth-Bendix completion algorithm. We discovered that the length of the normal form of $w \in \Sigma$ is at most 6 and the number of $|\Sigma^*/R_{E_2}|$ is 168.

We found a minimal equation set E_2 of the set of equations E_6 such that $|E_2| = 8$. On the other hand, we were able to construct a set of 8 equations $F_2 \not\subseteq E_6$ such that $\Sigma^*/R_{E_6} = \Sigma^*/R_{F_2}$. At the same time, we do not have any equation set E such that $\Sigma^*/R_{E_6} = \Sigma^*/R_E$ and $|E| < 8$.

In fact, we observed that the calculation time of the Knuth-Bendix completion algorithm does not necessarily decrease with the size of equation set, for certain parameters. The computation time of our implementation of the Knuth-Bendix completion algorithm takes 108 seconds for E_{all}, 221 seconds for E_6 and 757 seconds for E_5 (CPU Intel Xeon W3530 (2.80 GHz), Memory 12 GB). In this paper, we restricted the size of qubits, so as a future work,

Equation set	E_{all}	E_6	E_5
Computation time	108 s	221 s	757 s

we would like to investigate about 4 or more qubits quantum circuits. It is computationally expensive to run the Knuth-Bendix completion algorithm for 4 qubits quantum circuit rewriting systems. Our future work includes finding an efficient initial equation set for this algorithm to improve the computing cost. The minimal equation set for 3 qubits could be a good hint to construct an efficient initial equation set for 4 qubits.

Acknowledgements

I would like to thank Professor Yoshihiro Mizoguchi for his valuable advice and encouragement during the course of this study. I would like to thank Professor Setsuo Taniguchi and Progessor Hiroshi Yoshida for them valuable advice and encouragement during the course of this study. I am grateful to Professor Shuichi Inokushi and Professor Morozov Kirill for his useful suggestions. I am grateful to Professor Miguel A. Martin-Delgado for his helpful comments.

References

[Bar95] A. Barenco. A universal two-bit gate for quantum computation. *arXiv:quant-ph/9505016*, May 1995.

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

[Ben80] P. Benioff. The comuter as a physical system: A microscopic quantum mechanical hamiltonian model of comuputers as rep-

resented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

[Ben82] P. Benioff. Quantum mechanical models of turing machines that dissipate no energy. *Physical Review Letters*, 48(23):1581–1585, March 1982.

[BO81] R. V. Book and C. P. O'Dunlaing. Testing for the church-rosser property. *Theoritical Computer Science*, 16:223–229, 1981.

[BO93] R. V. Book and F. Otto. *String rewriting system*. Springer-Verlag, 1993.

[Boo82] R. V. Book. Confluent and other types of thue systems. *Jornal of the ACM*, 29:171–182, January 1982.

[Deu89] D. Deutsch. Quantum computational networks. *Series A-Mathematical and Physical Sciences.*, 425(1868):73–90, Sept 1989.

[DiV95] D. P. DiVincenzo. Two-bit gates are universal for quantum computation. *Physical Review A*, 51(2):1015–1022, 1995.

[Fey85] R. P. Feynman. Quantum mechanical computers. *Optics News*, 11:11–20, 1985.

[Gil79] R. H. Gilman. Presentations of group and monoids. *Journal of Algebra*, 57:544–554, 1979.

[GMD02] A. Galindo and M. A. Martin-Delgado. Information and computation: Classical and quantum aspects. *Reviews of Modern Physics*, 74:347–423, Applil 2002.

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search. *symposium on Theory of computing*, 28:212–219, 1996.

[Hue80] Gerard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27:797–821, 1980.

[IKY02] K. Iwama, Y. Kambayashi, and S. Yamashita. Transformation rules for designing CNOT-based quantum circuits. *Design Automation Conference*, 39th:419–424, 2002.

[KB70] D. Knuth and P. Bendix. Simple word problems in universal algebras. *Computational Problems in Abstract Algebra*, pages 263–297, 1970.

[KN85] D. Kapur and P. Narendran. A finite thue system with decidable word problem and without equivalent finite cononical system.

Theoritical Computer Science, pages 337–344, 1985.

[Kni95] E. Knill. Approximation by quantum circuits. *arXiv:quant-ph/9508006*, 1995.

[Met83] Y. Metivier. About the rewriting system produced by the knuth-bendix completion algorithm. *Information Processing Letters*, 16:31–34, 1983.

[NO88] P. Narendran and F. Otto. Elements of finite order for finite weight-reducing and confluent thue systems. *Acta Informatica*, 25(5):573–591, June 1988.

[OZ91] F. Otto and L. Zhang. Decision problems for finite special string-rewriting systems that are confluent on some congruence class. *Acta Informatica*, 28:477–50, 1991.

[PS04] D. W. Parkes and V. Yu. Shavrukov. Monoid presentations of groups by finite special string-rewritings systems. *RAIRO-Inf. Theor. Appl.*, 38:245–256, 2004.

[Sho94] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Foundations of Computer Science*, 35th:124–134, Nov 1994.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Tof81] Tommaso Toffoli. Bicontinuous extensions of inverrible combinatorial functions. *Mathematical Systems Theory*, 14:13–23, 1981.

[Wan98] J. Wang. Finite complete rewriting systems and finite derivtion type for small extensions of monoids. *Journal of Algebra*, 204:493–503, 1998.

[Yao93] A Chi-Chih Yao. Quantum circuit complexity. *Foundations of Computer Science*, 34th:352–361, 1993.

Appendix

We show the reslut KBA(E_{all}) of Kunth-Bendix algorithm for E_{all} and the monoid $\Sigma^*/R_{E_{all}}$.

KBA(E_{all}) = { $aa \to \lambda$, $abcae \to caed$, $abcfbd \to beabc$, $abcfd \to fbde$, $abd \to da$, $abea \to bef$, $abef \to bea$, $abf \to fb$, $acaed \to bcae$, $acfbd \to eabc$, $ada \to bd$, $adfcb \to dfcd$, $adfcd \to dfcb$, $aea \to ef$, $aef \to ea$, $afb \to bf$, $ba \to ab$, $bb \to \lambda$, $bcab \to cda$, $bcaeb \to afcda$, $bcaed \to acae$, $bcb \to cd$, $bcd \to cb$, $bceab \to adefd$, $bcead \to adefb$, $bda \to ad$, $bdea \to adef$, $bdef \to adea$, $bfb \to af$, $bfcb \to afcd$, $bfcd \to afcb$, $cabc \to acda$, $cabeb \to bebdf$, $cabed \to acabe$, $cac \to aca$, $cad \to bca$, $caebd \to fcda$, $caeda \to fbc$, $cafc \to acea$, $cbc \to bd$, $cbd \to bc$, $cbe \to bed$, $cbfc \to adea$, $cc \to \lambda$, $cdaeb \to ebdf$, $cdaf \to bcfb$, $cde \to ed$, $cdfc \to def$, $ceabc \to acbfd$, $cebd \to deb$, $ced \to de$, $cef \to fc$, $cfbc \to aeda$, $cfbde \to beabc$, $cfc \to ef$, $dab \to ad$, $dac \to acb$, $dad \to ab$, $daebd \to abceb$, $daed \to abce$, $dafcb \to bdfcd$, $dafcd \to bdfcb$, $db \to bd$, $dc \to cd$, $dd \to \lambda$, $deab \to cead$, $dead \to ceab$, $debd \to ceb$, $ded \to ce$, $dfb \to adf$, $dfcda \to aebdf$, $eabca \to acafd$, $eabce \to bcfbd$, $eabcf \to beabc$, $eabe \to befb$, $eac \to acf$, $eade \to afcd$, $eadf \to dfcb$, $eae \to af$, $eaf \to ae$, $ebc \to deb$, $ebde \to bceb$, $ebdfc \to bebdf$, $ebdfd \to aebdf$, $ebe \to beb$, $ebf \to aeb$, $ec \to ce$, $edae \to bcfb$, $edaf \to cdae$, ede, cd, $edf \to dfc$, $ee \to \lambda$, $efbc \to cfbd$, $efbd \to aeda$, $efc \to cf$, $fa \to af$, $fbca \to caed$, $fbce \to abcf$, $fbcf \to abce$, $fbdeb \to beabc$, $fbdf \to dafd$, $fbe \to bea$, $fbf \to ab$, $fca \to cae$, $fcbf \to ceab$, $fcdae \to aebdf$, $fcdf \to defd$, $fce \to cf$, $fcf \to ce$, $fda \to bfd$, $fde \to cfd$, $fdf \to dfd$, $fe \to ef$, $ff \to \lambda$ }

$\Sigma^*/R_{E_{all}} = \{\lambda,\ a,\ ab,\ abc,\ abca,\ abcaf,\ abcafd,\ abce,\ abcea,\ abceb,\ abcf,\ abcfb,\ abe,$ $abeb,\ abebd,\ abebdf,\ abed,\ abeda,\ ac,\ aca,\ acab,\ acabe,\ acae,\ acaeb,\ acaf,\ acafd,\ acb,\ acbf,$ $acbfd,\ acd,\ acda,\ acdae,\ acdf,\ acdfd,\ ace,\ acea,\ aceab,\ acead,\ aceb,\ acf,\ acfb,\ acfd,\ ad,$ $ade,\ adea,\ adeb,\ adef,\ adefb,\ adefd,\ adf,\ adfc,\ adfd,\ ae,\ aeb,\ aebd,\ aebdf,\ aed,\ aeda,\ af,$ $afc,\ afcb,\ afcd,\ afcda,\ afd,\ b,\ bc,\ bca,\ bcae,\ bcaf,\ bcafd,\ bce,\ bcea,\ bceb,\ bcf,\ bcfb,\ bcfbd,$ $bcfd,\ bd,\ bde,\ bdeb,\ bdf,\ bdfc,\ bdfcb,\ bdfcd,\ bdfd,\ be,\ bea,\ beab,\ beabc,\ bead,\ beb,\ bebd,\ bebdf,$ $bed,\ beda,\ bef,\ befb,\ befd,\ bf,\ bfc,\ bfd,\ c,\ ca,\ cab,\ cabe,\ cae,\ caeb,\ caed,\ caf,\ cafd,\ cb,$ $cbf,\ cbfd,\ cd,\ cda,\ cdae,\ cdf,\ cdfd,\ ce,\ cea,\ ceab,\ cead,\ ceb,\ cf,\ cfb,\ cfbd,\ cfd,\ d,\ da,\ dae,$ $daeb,\ daf,\ dafc,\ dafd,\ de,\ dea,\ deb,\ def,\ defb,\ defd,\ df,\ dfc,\ dfcb,\ dfcd,\ dfd,\ e,\ ea,\ eab,$ $eabc,\ ead,\ eb,\ ebd,\ ebdf,\ ed,\ eda,\ ef,\ efb,\ efd,\ f,\ fb,\ fbc,\ fbd,\ fbde,\ fc,\ fcb,\ fcd,\ fcda,$ $fd\}.$