ブール関数を計算する量子回路の局所変換ルールの完全集合

山下 茂 *[†], 岩間 一雄 *[‡]
* 科学技術振興事業団今井量子計算機構プロジェクト
[†]NTT コミュニケーション科学研究所
[‡] 京都大学大学院情報学研究科

要旨 本稿では，CNOT(制御 NOT) ゲートにより構成されるブール関数を計算する量子回路における，回路の局所変換ルールの集合について述べる．本稿で述べる局所変換ルールの集合は単純であるが自明ではない．また，その集合が完全であること，つまり，任意の二つの回路に対して一方から他方への変換が本稿で述べるルールのみを適用することによって可能であることも示す．

A Complete Set of Transformation Rules for Quantum Boolean Circuits

Shigeru Yamashita*[†] and Kazuo Iwama*[‡]
*Quantum Computation and Information, ERATO,
Japan Science and Technology Corporation (JST)
[†]NTT Communication Science Laboratories, Kyoto, Japan
[‡]School of Infomatics, Kyoto University, Kyoto, Japan

Abstract This paper gives a simple but nontrivial set of local transformation rules for CNOT-based quantum circuits. It is shown that this rule set is complete, namely for any two equivalent circuits, S_1 and S_2, there is a sequence of transformations, each of them in the rule set, which changes S_1 to S_2.

1 Introduction

For given Boolean formulas or circuits, S_1 and S_2, to determine whether or not S_1 is equivalent to S_2 is a fundamental coNP-complete problem. Note that coNP-completeness does not always deny the existence of short *witnesses* or *proofs*. In fact, there have been a lot of efforts to seek so-called *proof systems* which provide us (if any) with a short *proof*. Here a proof is a certificate with which we can verify $S_1 = S_2$ in polynomial time. Such systems include Resolution [e.g., Iwa97, Hak85, BEGJ98] and the more powerful Frege systems [e.g., CR79, PU95], both of which are used to prove that a given CNF formula is unsatisfiable (i.e., for the case that S_1 is a CNF formula and S_2 is the empty formula). Such a proof system is also known ([e.g., IHKS97]) for fan-in restricted combinatorial circuits. Although research efforts for these proof systems have been mostly devoted to obtain (exponential) lower-bounds in the theory community [IP95, Hak85, BEGJ98], their original motivation is obviously more positive, namely, to use them for automated theorem proving and circuit design. For example, DeMorgan's law is used very often to make a local simplification of Boolean circuits.

In this paper, we present a similar proof system for *quantum* Boolean circuits (QBCs). QBCs were introduced by Yao [Yao93] and have been popular as a standard model for describing quantum algorithms [e.g., Sho94] and quantum Boolean oracles [DJ92, Gro93]. [BBC$^+$95] shows that any unitary transformation can be broken down into a sequence of basic quantum gates. [LCKL99] gives a method of how to construct any QBC by using generalized Control-NOT (CNOT) gates, but an obtained circuit is like an elementary two-level AND/OR circuit in the classical design. Thus QBC's are well accepted as a standard model for quantum computation, just like classical Boolean circuits for classical computation, but their analysis is apparently at an early stage compared to their classical counterpart. In particular, we have no design theory to obtain desirable circuits while preserving the same functionality.

Our system in this paper consists of six transformation rules for CNOT-based quantum circuits. To prove their completeness, i.e., the existence of a sequence of transformations from any circuit to any other equivalent one, we introduce a unique canonical form of QBCs. Since each transformation is bidirectional, we can prove the completeness of the rule set by only showing that there is a sequence of transformations from any circuit to its canonical form. Using our system we can modify a given circuit into another with a desirable property (e.g., of small size) by executing a NP-type search, although the length of the search path might not be too short.

It should be noted that finding NP-type Transformation rules for coNP sets is apparently an interesting research topic, but few successful examples are known other than the previously mentioned proof systems for Boolean formulas. One rare but famous example is Hajós calculus for non-3-colorable graphs [IP95, Haj61, PU95], which consists of simple and beautiful rules. Our transformation rules are also simple and quite nontrivial, each of which is carried out in polynomial time.

2 Quantum Boolean Circuits

Intuitively, a quantum Boolean circuit is given as illustrated in Fig.2. It is a quantum system with $n+1$ *qubits* (*quantum bits*), denoted as $|x_1\rangle|x_2\rangle\cdots|x_{n+1}\rangle$. As an interaction of their qubits, we can only use control-NOT (CNOT) gates whose functionality will be given later. The input and the corresponding output states of the first n qubits have to be identical in our model. The $(n+1)$-st qubit state $|x_{n+1}\rangle$ is changed to $|x_{n+1} \oplus f(x_1, \cdots, x_n)\rangle$, and is considered as a special bit, so called a *work bit*, which is used to obtain the value of the Boolean function f for inputs $x_1, \cdots, x_n$. Furthermore, a circuit can use any (finite) number of *auxiliary qubits* which are reset to $|0\rangle$ initially. The reason why we introduce auxiliary qubits is as follows: We can sometimes decrease dramatically the number of required gates by using auxiliary qubits although auxiliary qubits require some implementation costs. Therefore, there is a trade-off between the number of auxiliary qubits and the number of CNOT gates. Our circuit model can accommodate both design strategies, i.e., the strategy for saving qubits and the strategy for saving CNOT gates.

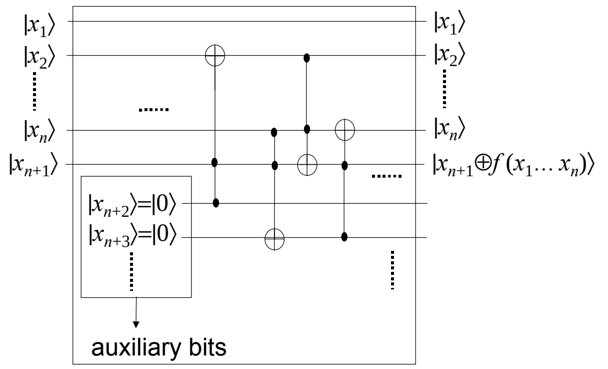

Figure 1: A Quantum Boolean Circuit

More formally, a *quantum Boolean circuit* is given as a sequence $[t_1, C_1] \cdot [t_2, C_2] \cdots [t_m, C_m]$, where (i) t_i is an integer ≥ 1 and (ii) C_i is a finite set of integers ≥ 1. Each $[t_i, C_i]$ is a Control-NOT (CNOT) gate whose control bits are given by C_i and its target bit is given by t_i. (CNOT gates are sometimes called Toffoli gates as in [Gru99].) For example, the leftmost CNOT gate of Fig. 2 is given by $[2, \{n+1, n+2\}]$, namely, if $t_i \leq n+1$, then it shows x_{t_i}, and if $t_i > n+1$, then it shows the $(t_i - n - 1)$-th auxiliary qubit. We can use finitely many auxiliary qubits, which are denoted by $|x_{n+2}\rangle \cdots |x_N\rangle$. Note that, for example, the state of first qubit may change after a CNOT gate whose target bit is the first qubit. However, we often say "qubit $|x_1\rangle$" to show the qubit itself regardless of its current state.

We denote the set of n-bit basis vectors by $\{0,1\}^n$. The sate of n qubits $|x_1\rangle \cdots |x_n\rangle$ is a superposition (a linear combination) of those 2^n basis vectors. However, we often assume in this paper that the state is a single basis vector when describing the behavior of the system. Generalization to superposed states can be done simply by taking a linear combination of the results for each basis vector. The following definitions of our circuits and its internal states follow this convention: Let $[t_1, C_1] \cdots [t_i, C_i]$ be a prefix of the circuit $[t_1, C_1] \cdots [t_i, C_i] \cdots [t_m, C_m]$. Then we define the sate S_i of the quantum system as follow:

(1) $S_0 = |a_1\rangle|a_2\rangle \cdots |a_{n+1}\rangle|0\rangle|0\rangle \cdots |0\rangle$, where $|a_1\rangle|a_2\rangle \cdots |a_{n+1}\rangle \in \{0,1\}^n$ is an input state.

(2) Suppose that $S_i = |x_1^i\rangle|x_2^i\rangle \cdots |x_{n+1}^i\rangle|x_{n+2}^i\rangle \cdots |x_N^i\rangle$, and $S_{i+1} = |x_1^{i+1}\rangle|x_2^{i+1}\rangle \cdots |x_{n+1}^{i+1}\rangle|x_{n+2}^{i+1}\rangle \cdots |x_N^{i+1}\rangle$. Then $|x_k^{i+1}\rangle = |x_k^i\rangle$ if $k \neq t_i$, and $|x_{t_i}^{i+1}\rangle = |x_{t_i}^i \oplus X_{C_i}^k\rangle$ where $X_{C_i}^k$ is a product term of all x_j^i such that $j \in C_i$. For example, we can calculate S_1 of Circuit A in Fig. 2 such that $|x_i^1\rangle = |x_i\rangle$ for $i \neq 6$, and $|x_6^1\rangle = |x_6 \oplus x_1 \cdot x_3 \cdot x_7\rangle$.

Definition 1. Let $(a_1, a_2, \cdots a_n, a_{n+1}) \in \{0,1\}^n$, and suppose that the initial state $S_0 = |a_1\rangle|a_2\rangle \cdots |a_{n+1}\rangle|0\rangle \cdots |0\rangle$. Also suppose that the final state $S_m = |b_1\rangle|b_2\rangle \cdots |b_{n+1}\rangle|b_{n+2}\rangle \cdots |b_N\rangle$. Then the circuit is said to be *proper* if $|b_k\rangle = |a_k\rangle$ for $1 \leq k \leq n$, $|b_k\rangle = |0\rangle$ for $n+2 \leq k \leq N$ and $|b_{n+1}\rangle$ is equal to $|a_{n+1} \oplus f(a_1, \cdots, a_n)\rangle$. It is also said the circuit *computes* a Boolean function $f(x_1, \cdots, x_n)$. Note that if the auxiliary qubits are not reset to $|0\rangle$ at the end of a circuit, we cannot use the circuit as a Boolean oracles. The reason is that operations which interact some of the states of $|x_1\rangle \cdots |x_n\rangle$ with each other, such as Hadamard transormation, do not work as desired if the states of $|x_1\rangle \cdots |x_n\rangle$ are entangled with the sates of other qubits.

It should be noted that many existing Boolean oracles are proper and they need to be so. For example, the *f-controlled phase shift*, U, used in the Grover's search algorithm [Gro93] is defined as $|x_1\rangle \cdots |x_n\rangle|x_{n+1}\rangle = (-1)^{f(x_1,\cdots,x_n)}|x_1\rangle \cdots |x_n\rangle|x_{n+1}\rangle$ where $|x_{n+1}\rangle$ is initialized to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. If the circuit is not proper, we can no longer use the circuit as a primitive of the above unitary operation.

Fig. 2 shows three quantum circuits which are all equivalent, i.e., they compute the same Boolean fucntion $f(x_1, \cdots, x_6) = (x_1 \oplus x_2) \cdot (x_3 \oplus x_4) \cdot (x_5 \oplus x_6)$. Note that x_7 is a work bit in these circuits. As will be shown later, we can construc a circuit only using CNOT gates whose target bit is x_{n+1} like Circuit B. However, different types of circuits are of course possible like Circuits A and C, where C is simpler than A and B.

3 Transformation Rules

In this section, we introduce six *transformation rules* which can be applied for a sequence of CNOT gates. Each transformation rule looks like $F \Leftrightarrow G$, which means that we can transform F to G, and vice versa, where F and G are sequences of CNOT gates.

Transformation Rule Set. In the followings, ϵ means the empty sequence, and we refer to a CNOT gate whose target bit is the i-th bit as $CNOT_i$.

(1) $[t_1, C_1] \cdot [t_1, C_1] \Leftrightarrow \epsilon$.

(2) $[t_1, C_1] \cdot [t_2, C_2] \Leftrightarrow [t_2, C_2] \cdot [t_1, C_2]$, if $t_1 \notin C_2$ and $t_2 \notin C_1$. (The condition means that the two gates are "independent." If there is some influence between two gates, we cannot change the order of the two gates. Even in such cases, we can change the order by adding some gates, as we will see in the following Transformation Rules.)

(3) $[t_1, C_1] \cdot [t_2, C_2] \Leftrightarrow [t_2, C_2] \cdot [t_1, C_1] \cdot [t_1, C_1 \cup C_2 - \{t_2\}]$, if $t_1 \notin C_2$ and $t_2 \in C_1$.

(4) $[t_1, C_1] \cdot [t_2, C_2] \Leftrightarrow [t_2, C_1 \cup C_2 - \{t_1\}] \cdot [t_2, C_2] \cdot [t_1, C_2]$, if $t_1 \in C_2$ and $t_2 \notin C_1$. (This rule is a dual case of (3), i.e., the relationship between the two gates is just opposite.)

(5) $[t_1, \{c_1\}] \cdot [t_2, C_2 \cup \{c_1\}] \Leftrightarrow [t_1, \{c_1\}] \cdot [t_2, C_2 \cup \{t_1\}]$, if $(t_1 > n+1)$ and there is no $CNOT_{t_1}$ before $[t_1, \{c_1\}]$.

(6) $[t, C] \Leftrightarrow \epsilon$, if there is a integer i such that $i \in C, i > n+1$, and there is no $CNOT_i$ before

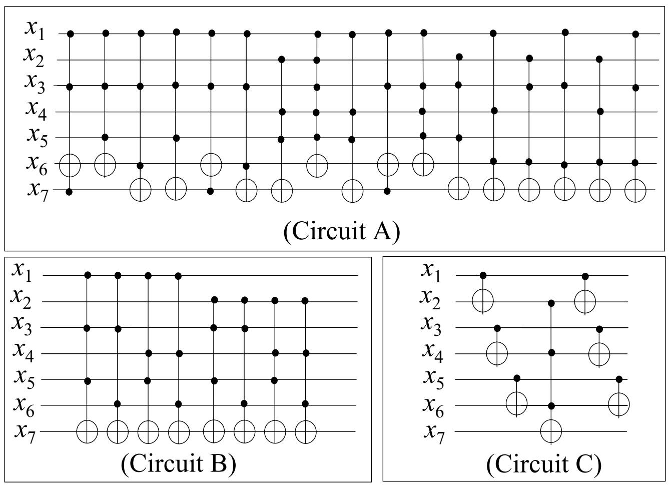

Figure 2: An Example of Equivalent Circuits

$[t, C]$.

Lemma 1. Applying any one of Rules (1) to (6) does not change the Boolean function the circuit computes.

Proof. (1) Rule (1) is obvious since $f \oplus f = 0$ for any Boolean function f.

(2) Rule (2) is also easy because $[t_1, C_1]$ and $[t_2, C_2]$ do not affect each other if $t_1 \notin C_2$ and $t_2 \notin C_1$.

(3) Fig.3 shows this transformation. We have two important qubits, i.e., the qubits whose indices are t_1 and t_2 ($t_1 \neq t_2$ by the condition). Let a_1 and a_2 be the states of $|x_{t_1}\rangle$ before and after the gate $[t_1, C_1]$, respectively, in the lefthand-side circuit. Let a_1, a'_2 and a'_3 be the states of $|x_{t_1}\rangle$ before $[t_2, C_2]$, after $[t_1, C_1]$ and after $[t_1, C_1 \cup C_2 - \{t_2\}]$, respectively, in the righthand-side circuit. Also, b_1, b_2 and b'_2 are the states of $|x_{t_2}\rangle$ similarly defined (see Fig.3). Furthermore, let A and B be the conjunction of the qubits whose indices are contained in C_2 and $C_1 - \{t_2\}$, respectively (A and B may include common qubits). Note that the sates whose indices are in A or B do not change throughout this portion of the circuit since there is no target bit whose index is in A or B. Now let us calculate the states of a_2, b_2, a'_3 and b'_2. First, both b_2 and b'_2 can be written as $b_1 \oplus A$. Furthermore, $a_2 = a_1 \oplus B \cdot b_1$, $a'_2 = a_1 \oplus B \cdot b'_2 = a_1 \oplus B \cdot (b_1 \oplus A) = a_1 \oplus B \cdot b_1 \oplus A \cdot B$ and $a'_3 = a'_2 \oplus A \cdot B = a_1 \oplus B \cdot b_1 \oplus A \cdot B \oplus A \cdot B = a_1 \oplus B \cdot b_1$. Thus we have shown that $b_2 = b'_2$ and $a_2 = a'_3$. The states of other qubits do not change obviously and hence the transformation does not change the functionality of the circuit.

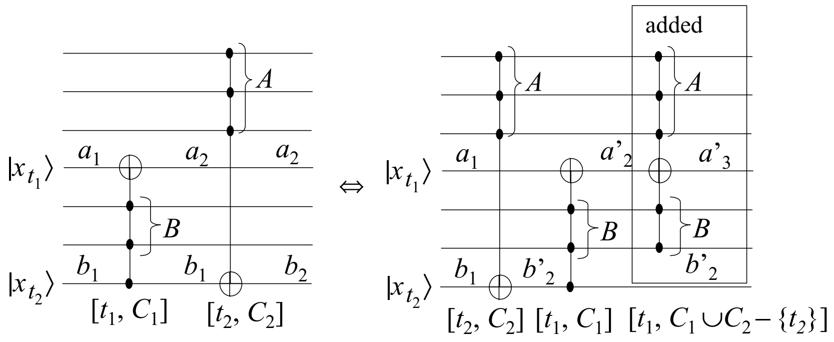

Figure 3: An Example of Transformation Rule 3

(4) Similar to the proof for Rule (3).

(5) If there is no $CNOT_{t_1}$ before $[t_1, \{c_1\}]$, the state of the t_1-th state remains as $|0\rangle$ just before $[t_1, \{c_1\}]$. Therefore, after applying $[t_1, \{c_1\}]$, the t_1-th bit and the c_1-th bit can be regarded as the same, which means the rule $[t_2, C_2 \cup \{c_1\}] \Leftrightarrow [t_2, C_2 \cup \{t_1\}]$ does not change functionality.

(6) If there is an integer i which satisfies the condition of the rule, $[t, C]$ has an auxiliary bit $|x_i\rangle$ as its control bit whose current sate remains as $|0\rangle$. This means that the conjunction by C is also always 0 and this gate is completely useless. $\Box$

4 Completeness of the Rule Set

In this section, we first introduce the canonical form for quantum Boolean circuits. Then it is shown that any circuit can be transformed into its canonical form using the transformation rules. The completeness of the rule set is its immediate consequence.

4.1 Canonical Form

Definition 2. A quantum Boolean circuit S is said to be of the canonical form, if (1) it has only $CNOT_{n+1}$ gates, which denote CNOT gates whose target bit is the work bit, (2) it does not have two or more same $CNOT_{n+1}$ gates, (3) $CNOT_{n+1}$ gates are ordered lexicographically in terms of the indices of their control bits, and (4) no auxiliary qubits are used.

The condition (2) means that the canonical form must not be redundant in terms of Transformation Rule 1. The condition (3) means that, for example, $[n+1, \{1, 2, 3\}]$ should be placed before $[n, \{2, 3, 4\}]$. Recall that Fig. 2 shows three circuits computing the same Boolean function, and only Circuit B is of the canonical form. Now here is an important lemma:

Consider the following form of Boolean formulas:

$$a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \oplus a_{1,2} x_1 x_2 \oplus a_{1,3} x_1 x_3 \oplus \cdots \oplus a_{n-1,n} x_{n-1} x_n \oplus \cdots \oplus a_{1,2,\cdots n-1,n} x_1 x_2 \cdots x_{n-1} x_n.$$

Note that there are no negative literals in the formula and each a_i is 0 or 1. This form is called a *positive polarity Reed-Muller expression* [DDT78].

Lemma 2 [DDT78]. Any Boolean function can be expressed by a *positive polarity Reed-Muller expression* which is unique except for the order of terms.

Lemma 3. Any CNOT-based quantum circuit S has its unique canonical form.

Proof. By definition, S transforms $|x_i\rangle$ to $|x_i\rangle$ for $1 \le i \le n$ and $|x_{n+1}\rangle$ to $|x_{n+1} \oplus f(x_1, \cdots, x_n)\rangle$ for some formula f. Compute the positive polarity Reed-Muller expression of f, which naturally corresponds to a sequence of $CNOT_{n+1}$ gates. Since the order of conjunctive terms must be lexicographic, the resulting circuit is unique by Lemma 2. $\Box$

4.2 Transformation Procedure

Theorem 1. Any quantum circuit S can be transformed into its canonical form using the transformation rules.

Proof. Our procedure of such a transformation is given in Figs. 4 and 5. We first call $Main(S)$. Then it calls $Shift(S, n+1)$ which "shifts" all the $CNOT_{n+1}$ gates to the left part. In $Shift$, $MostLeft(S,$ "a statement$'')$ denotes the most left gate which satisfies the statement in circuit S, and $Left(S, f)$ denotes the part of S which is the left-hand side of f. *MostRight* and *Right* have similar meanings.

Then $Main$ calls $Shift(S', n)$, by which all $CNOT_n$ gates are shifted to the left. They are placed next to $CNOT_{n+1}$ gates already shifted. This continues until $Shift(S', 1)$.

Now we execute the second half of $Main$, where all the gates whose target bits are not $n+1$ are deleted, and there remain only $CNOT_{n+1}$ gates. One might be curious why that happens. The reason is simple: Suppose that some of them, say some $CNOT_n$ gate does not disappear after applying Rules (1), (2), and (6) (note that any two $CNOT_n$ gates can be exchanged by Rule (2)) as many times as possible. Then the output state of the n-th qubit must be different from its input state $|x_n\rangle$, which can be easily proved by the property of the positive polarity Reed-Muller expression. This violates our definition of proper circuits, which means that the original circuit is not proper. □

4.3 Completeness of the Rule Set

Now our main result is almost immediate:

Theorem 2. Let S_1 and S_2 be any equivalent quantum Boolean circuits. Then there exists a sequence of transformation rules, each in the rule set given in Section 3, which transforms S_1 to S_2.

Proof. Since S_1 and S_2 are equivalent, Lemma 3 tells us that their canonical forms must be the same. Let this canonical form be S. We can transform S_1 into its canonical-form circuit S by Theorem 1. Let this sequence of transformation rules be r_1. We can also transform S_2 into S by sequence r_2. Since all of our transformation rules are bidirectional, we can get a sequence r_2' of rules that transforms C to C_2 just by reversing r_2. Now the sequence r_1 followed by r_2' transforms C_1 into C_2. □

5 Concluding Remarks

Apparently several research topics are remaining: (1) Although we give a complete set of transformation rules, it is open how to apply them in order to get a *short* sequence of transformation. The sequence described in Theorem 2 is apparently not efficient. (2) We must have some concrete goal when applying transformation rules such as simplifying the circuit. Is there some strategy to find a sequence of transformations which gives us a better circuit? (3) It is also interesting to develop a design theory for *sequential* quantum circuits which include registers, by exploiting our design theory for combinatorial quantum circuits.

Acknowledgment

The authors are grateful to Mr. Hirotada Kobayashi for his useful comments on the properness of our circuit model.

References

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

[BEGJ98] M. Bonet, J. Esteban, N. Galesi, and J. Johannsen. Exponential separation between ressticted resolution and cutting planes proof systems. In *FOCS'98*, pages 638–647, 1998.

[CR79] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. In *J. Symbolic Logic, 44*, pages 36–50, 1979.

```
$Shift(S,i)$
{
    $f = MostLeft(S,$ "$1 \leq$ target bit $\leq i-1$")
    while (gate $g = MostLeft((Right(S,f),$ "target bit is $i$") exists) {
        $h =$ the gate just before $g$
        let $h = [t_1, C_1]$ and $g = [i, C_2]$
        if ($t_1 \in C_2$ and $i \in C_1$) {
            let the first unused auxiliary qubit be $t$-th bit
            add two gate $k_1 = [t, \{i\}]$ and $k_2 = [t, \{i\}]$ before $h$ by Transformation Rule 1
            change the order of $k_2$ and $h$ by Transformation Rule 2
            change $h = [t_1, C_1' \cup \{i\}]$ to $h' = [t_1, C_1' \cup \{t\}]$ by applying Transformation Rule 5 to $k_1$ and $h$
            let $S_1 = Left(S,h)$ and $S_2 = Right(S,h)$
            $f_1 = MostLeft(S_1,$ "target bit is not $t$")
            while (gate $g_1 = MostLeft((Right(S_1, f_1),$ " target bit is $t$") exists) {
                $h_1 =$ the gate just before $g_1$
                change the order of $h_1$ and $g_1$ by either one of Transformation Rules 2 or 4
                  /* it is always possible */
            }
            $f_2 = MostRight(S_2,$ "$1 \leq$ target bit $\leq i$")
            while (gate $g_2 = MostRight((Left(S_2, f_2),$ "target bit is $t$") exists) {
                $h_2 =$ the gate just after $g_2$
                change the order of $g_2$ and $h_2$ by either one of Transformation Rules 2 or 3
                  /* it is always possible */
            }
        }
        /* now, we can always apply one of Transformation Rules 2 to 4 */
        change the order of $h$ and $g$ by one of Transformation Rules 2 to 4
    }
    $S_3 = Left(S, MostLeft(S,$"target bit is $i$"))
    $f_3 = MostLeft(S_3,$"the $i$-th bit is a control bit")
    let $f_3$ be $[t_1, \{i\}]$
    move $f_3$ to the leftmost in $S_3$ by applying Transformation Rule 2 /* it is always possible */
    $S_4 = Right(S_3, f_3)$
    while (gate $g_3 = MostLeft((S_4,$"the $i$-th bit is a control bit") exists) {
        if ($g_3$ is not the leftmost gate in $S_4$) {
            move $g_3$ to the leftmost in $S_4$ by applying Transformation Rule 2/* it is always possible */
            change $g_3 = [t_2, \{i\}]$ to $g_3' = [t_2, \{t_1\}]$ by applying Transformation Rule 6 to $f_3$ and $g_3$
        }
    }
    $f_4 = MostRight(S,$ "$1 \leq$ target bit $\leq i$")
    while (gate $g_4 =$
        $MostRight((Left(S, f_4),$"the $i$-th bit is neither the target bit nor a control bit") exists) {
        $h_4$ =the gate just after $g_4$
        change the order of $g_4$ and $h_4$ by either one of Transformation Rules 2 or 4
    }
    $f_5 = MostLeft(S,$"target bit is $i$ and $t_1$ is not a control bit")
    while (gate $g_5 = MostLeft((Right(S, f_5),$ "target bit is $i$ and $t_1$ is a control bit") exists) {
        $h_5 =$ the gate just before $g_5$
        change the order of $h_5$ and $g_5$ by Transformation Rule 2
    }
    $S_5 = Right(Left(S, MostLeft(S,$"target bit is $i$ and $t_1$ is not a control bit"$), f_3)$
    order $CNOT_i$ lexicographically in $S_5$ by applying Transformation Rules 2
    delete two adjacent gates in $S_5$ if they are the same by applying Transformation Rules 1
    /* by the above transformation, $S_5$ must be changed to $\epsilon$ because $x_i \oplus x_i \cdot G(X) = x_i \oplus F(X)$ holds
        only when $G(X) = 0$ for any Boolean function $F$ and $G$,
        where $X$ is a variables set which does not contain $x_i$ */
    while (gate $g_6 = MostLeft((Right(S, f_3),$ "$1 \leq$ target bit $\leq i$") exists) {
        $h_6 =$ the gate just after $f_3$
        change the order of $f_3$ and $h_6$ by one of Transformation Rules 2 to 4
    }
}
```

Figure 4: $Shift(S,i)$

```
Main(S)
{
    S' = S
    for ($i = n + 1$ to 1) {
        $Shift(S', i)$
        $C_i = S_L$ /* the left part of the result of $Shift(S', i)$ which consist of only $CNOT_i$ */
        $S' = S_R$ /* the right part of the result of $Shift(S', i)$ */
        delete redundant gates in $C_i$ by applying Transformation Rule 6
    }
    order gates lexicographically by applying Transformation Rules 2
    delete two adjacent gates if they are the same by applying Transformation Rules 1
    /* by the above transformation, only $CNOT_i$ gates remain */
}
```

Figure 5: $Main(S)$

[DDT78] M. Davio, J-P. Deshamps, and A. Thayse. *Discrete and Switching Functions.* McGraw Hill International, 1978.

[DJ92] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. In *Proceedings of the Royal Society London A*, pages 439:553–558, 1992.

[Gro93] L. K. Grover. A fast quantum mechanical algorithm for database search. In *STOC' 96*, pages 212–219, 1993.

[Gru99] J. Gruska. *Quantum Computing.* McGraw Hill, 1999.

[Haj61] G. Hajós. Über eine Konstruktion nicht n-färbbarer Graphen. In *Wiss. Z. Martin-Luther-Univ, Halle-Wittenberg, xA10*, pages 116–117, 1961.

[Hak85] A. Haken. The intractability of resolution. In *Theoretical Computer Science, 39*, pages 297–308, 1985.

[Iwa97] K. Iwama. Complexity of Finding Short Resolution Proofs. In *Proc. 22nd Symposium on Mathematical Foundation of Computer Sceince (MFCS'97), LNCS 1295*, pages 309–318, 1997.

[IHKS97] K. Iwama, K. Hino, H. Kurokawa, and S. Sawada. Random benchmark circuits with controlled attributes. In *Proc. European Design & Test Conference and Exhibition (ED&TC'97)*, pages 90–97, 1997.

[IP95] K. Iwama and T. Pitassi. Exponential lower bounds for the tree-like Hajós calculu. In *Inform. Process. Lett. 54*, pages 289–294, August 1995.

[LCKL99] J-S. Lee, Y. Chung, J. Kim, and S. Lee. A Practical Method of Constructing Quantum Combinational Logic Circuits. Technical Report quant-ph/9911053, LANL e-print, 1999.

[PU95] T. Pitassi and A. Urquhart. The complexity of the Hajós calculus. In *SIAM J. Discrete Math., 8*, pages 464–483, 1995.

[Sho94] Peter W. Shor. Algorithms for quantum computation: discrete log and factoring. In *FOCS' 94*, pages 124–134, 1994.

[Yao93] A. Yao. Quantum Circuit Complexity. In *FOCS' 93*, pages 352–361, 1993.