Vol. 46 No. 10 IPSJ Journal Oct. 2005

Regular Paper

Quantum versus Classical Pushdown Automata in Exact Computation

Yumiko Murakami,† Masaki Nakanishi,† Shigeru Yamashita†
and Katsumasa Watanabe†

Even though quantum computation is useful for solving certain problems, classical computation is more powerful in some cases. Thus, it is significant to compare the abilities of quantum computation and its classical counterpart, based on such a simple computation model as automata. In this paper we focus on the quantum pushdown automata which were defined by Golovkins in 2000, who showed that the class of languages recognized by quantum pushdown automata properly contains the class of languages recognized by finite automata. However, no one knows the entire relationship between the recognitive abilities of quantum and classical pushdown automata. As a part, we show a proposition that quantum pushdown automata can deterministically solve a certain problem that cannot be solved by any deterministic pushdown automata.

1. Introduction

In computational model theory, some quantum counterparts of classical computational models, such as quantum finite automata, have been proposed. These restricted quantum Turing machines may tell us the essential power of quantum computation, that is, the gap between quantum and classical computations. In particular the considerations of quantum finite automata [1),2)] and quantum counter automata [3)~6)] are conspicuous. In general, the quatum computational model is considered to be stronger than the classical one, however, the power of the quantum computation varies according to the setting, e.g., 1 way quantum finite automata are properly weaker than 1 way classical finite automata [1)]. In this paper, we focus on quantum pushdown automata, QPAs.

QPAs were introduced by C. Moore and J.P. Crutchfield [9)], but the authors were actually dealing with generalized QPAs whose evolution does not have to be unitary. Thus, M. Golovkins [10)] reintroduced QPAs including unitarity criteria. The author showed that the class of languages recognized by finite automata is properly contained in the class of languages recognized by QPAs, and further that QPAs can recognize some languages that cannot be recognized by deterministic pushdown automata, DPAs. Specifically, QPAs can recognize:

- every regular language with probability 1;
- a non-regular language $L_{a=b} = \{\omega \in (a,b)^* \mid |\omega|_a = |\omega|_b\}$ with probability 1;
- a non-context-free language $L_{a=b=c} = \{\omega \in (a,b,c)^* \mid |\omega|_a = |\omega|_b = |\omega|_c\}$ with probability 2/3; and
- a non-context-free language $L_{xor} = \{\omega \in (a,b,c)^* \mid |\omega|_a = |\omega|_b \text{ xor } |\omega|_{\mathrm{a}} = |\omega|_{\mathrm{c}}\}$ with probability 4/7,

where $|\omega|_a$ denotes the number of occurrences of a in the string ω.

His results show that bounded-error QPAs can be more powerful than DPAs. It remained open whether QPAs can be more powerful than classical pushdown automata in a fair setting, i.e., both in a bounded-error setting or both in a deterministic setting. In this paper, we answer the latter case affirmatively, that is, QPAs can be more powerful in a deterministic case. We show that there exists a problem that can be solved by QPAs deterministically, but cannot be solved by DPAs. This is the strict gap between the power of QPAs and their classical counterpart. The problem is a promise problem and cannot be directly related to a language. Instead, we use the setting that there are "acceptable", "rejectable", and "don't care" inputs, and discuss whether the acceptable and the rejectable are correctly recognized. Promise problems are discussed to show gaps of the power of quantum Turing machines. The Deutch-Jozsa promise problem [11)] and Simon's problem [13)], for example. And also sometimes, in the automaton model [15)] and communication complexity [16),17)].

Our main idea utilizes the Deutsch-Jozsa algorithm [11)] to construct a QPA that can solve the problem. Pop operations, which delete the

† Nara Institute of Science and Technology

stack top symbol, are restricted in QPAs since delete operations are not unitary in general. Therefore, it is not trivial to employ the algorithm. For the proof that the problem cannot be solved by DPAs, we utilize the generalized Ogden's lemma [14]. It should be noted that we need to modify the lemma for our purpose since it is a promise problem.

This paper is organized as follows: following this introduction, Section 2 defines QPAs, their configuration and behavior. Section 3 defines a certain problem and constructs a QPA that can deterministically solve it. Section 4 introduces the generalized Ogden's lemma and shows that there are no DPAs that solve the problem. Finally, Section 5 describes conclusions and future outlooks.

2. Preliminaries

2.1 Definitions

We cite the definition of QPAs, which are called simplified QPAs, from Ref. 10). "Simplified" means that the moving directions of the input tape head are always related to the next visiting states. We also cite the definition of their configuration and evolution.

Definition 2.1. *A Quantum Pushdown Automaton, QPA, is defined as the following 8-tuple.* $A=(Q,\Sigma,T,q_0,Q_{acc},Q_{rej},D,\delta)$ *is specified by a finite set of states* Q*, a finite input alphabet* Σ*, a finite stack alphabet* T*, an initial state* $q_0 \in Q$*, sets* $Q_{acc} \subset Q$*,* $Q_{rej} \subset Q$ *of accepting and rejecting states, respectively, with* $Q_{acc} \cap Q_{rej} = \phi$*, a function* $D : Q \longrightarrow \{\downarrow, \rightarrow\}$*, where* $\{\downarrow, \rightarrow\}$ *is the set of directions of input tape head, remaining at the current position or moving one cell forward, and a transition function* $\delta : Q \times \Gamma \times \Delta \times Q \times \Delta^* \longrightarrow \mathbf{C}$*, where* $\Gamma = \Sigma \cup \{\#, \$\}$ *is the input tape alphabet of* A *and* $\#, \$$ *are endmarkers not in* Σ*,* $\Delta = T \cup \{z\}$ *is the working stack alphabet of* A*, and* $z \notin T$ *is the stack bottom symbol.* □

The transition function is restricted to the following requirement:
If $\delta(q,\alpha,\beta,q',\tau) \neq 0$, then
(1) $|\tau| \leq 2$, and
(2) $\tau \in \beta T^*$ if $|\tau| \neq 0$.

Definition 2.2. *The configuration of a QPA is denoted as* $|c\rangle = |\nu_i q_j \nu_k, \tau_l\rangle$*, where the automaton is in a state* $q_j \in Q$*,* $\nu_i\nu_k \in \#\Sigma^*\$$ *is a finite word on the input tape,* $\tau_l \in zT^*$ *is a finite word on the stack tape, the input tape head is above the first alphabet of the word* ν_k*, and the stack head is above the last alphabet of the word* τ_l*.* □

Let C be the set of all configurations of a QPA. Set C is countably infinite. Since every configuration $|c\rangle$ denotes a basis vector in Hilbert space $H_A = l_2(C)$, a global state of A in space H_A has a form $|\psi\rangle = \sum_{c\in C} \alpha_c |c\rangle$, where $\alpha_c \in \mathbf{C}$ denotes the probability amplitude of a configuration $|c\rangle$, and $\sum_{c\in C} |\alpha_c|^2 = 1$.

Definition 2.3. *Let* $|c\rangle = |\nu_i q_j \sigma \nu_k, \tau_l \tau\rangle$*. A linear operator* U_A *is defined as follows:*

$$U_A|c\rangle = \sum_{(q,\tau')\in Q\times\{\varepsilon,\Delta,\Delta^2\}} \delta(q_j,\sigma,\tau,q,\tau')\,|f(|c\rangle,q),\tau_l\tau'\rangle,$$

where $f(|\nu_i q_j \sigma \nu_k, \tau_l\tau\rangle, q)$

$$= \begin{cases} \nu_i q \sigma \nu_k, & if\ D(q) = `\downarrow' \\ \nu_i \sigma q \nu_k, & if\ D(q) = `\rightarrow'. \end{cases}$$ □

For QPA $A = (Q,\Sigma,T,q_0,Q_{acc},Q_{rej},D,\delta)$, we define $C_{acc} = \{|\nu_i q_j \nu_k, \tau_l\rangle \in C | q_j \in Q_{acc}\}$, $C_{rej} = \{|\nu_i q_j \nu_k, \tau_l\rangle \in C \mid q_j \in Q_{rej}\}$, and $C_{non} = C \setminus (C_{acc} \cup C_{rej})$. E_{acc}, E_{rej}, and E_{non} are subspaces of H_A spanned by C_{acc}, C_{rej}, and C_{non}, respectively. We use the observable $\mathcal{O}$ that corresponds to the orthogonal decomposition $H_A = E_{acc} \oplus E_{rej} \oplus E_{non}$. The outcome of each measurement is either "accept" or "reject" or "non-halting."

The computation of QPA A proceeds as follows. For an input $\omega \in \Sigma^*$ we assume that computation starts with configuration $|q_0\#\omega\$, z\rangle$. Each computation step consists of two parts. First, linear operator U_A is applied to the current state, and then the resulting superposition is measured with respect to the observable $\mathcal{O}$ defined above. Let the state before the measurement be $\sum_{c\in C} \alpha_c |c\rangle$, and then the probability that the resulting superposition is projected into subspace E_i, $i \in \{acc, rej, non\}$, is $\sum_{c\in C_i} |\alpha_c|^2$. Computation continues until the result of a measurement is "accept" or "reject."

A QPA is considered valid in terms of quantum theory if its evolution operator is unitary.

Well-formedness conditions.
(1) $\forall(q_1,\sigma_1,\tau_1) \in Q\times\Gamma\times\Delta$,
$$\sum_{(q,\omega)\in Q\times\Delta^*} |\delta^*(q_1,\sigma_1,\tau_1,q,\omega)|^2 = 1.$$
(2) For all triples $(q_1,\sigma_1,\tau_1) \neq (q_2,\sigma_1,\tau_2)$ in

$Q \times \Gamma \times \Delta$,
$$\sum_{(q,\omega)\in Q\times\Delta^*} \delta^*(q_1,\sigma_1,\tau_1,q,\omega)\times \delta(q_2,\sigma_1,\tau_2,q,\omega) = 0.$$

(3) $\forall(q_1,\sigma_1,\tau_1,\tau_2) \in Q \times \Gamma \times \Delta^2$,
$$\sum_{(q,\tau,\omega)\in Q\times\Delta\times\{\varepsilon,\tau_2,\tau_1\tau_2\}} |\delta(q,\sigma_1,\tau,q_1,\omega)|^2 = 1.$$

(4) $\forall(q_1,\sigma_1,\tau_1),(q_2,\sigma_1,\tau_2)\in Q\times\Gamma\times\Delta, \forall\tau_3\in\Delta$,

(a) $$\sum_{(q,\tau)\in Q\times\Delta} \delta^*(q_1,\sigma_1,\tau_1,q,\tau) \times \delta(q_2,\sigma_1,\tau_2,q,\tau_3\tau) + \sum_{q\in Q} \delta^*(q_1,\sigma_1,\tau_1,q,\varepsilon) \times \delta(q_2,\sigma_1,\tau_2,q,\tau_3) = 0,$$

(b) $$\sum_{q\in Q} \delta^*(q_1,\sigma_1,\tau_1,q,\varepsilon) \times \delta(q_2,\sigma_1,\tau_2,q,\tau_2\tau_3) = 0.$$

Theorem 2.1. *Well-formedness conditions are satisfied iff evolution operator U_A is unitary.*

Proof. See the literature [10]. □

Throughout this paper, we consider only unitary QPAs that satisfy Well-formedness conditions.

3. QPAs That Solve a Certain Problem Deterministically

In this section, we show that QPAs can solve the following problem deterministically.

Problem I
[Input] A string $\omega = x\%y\%z\%y'\%z'$, where % is a separator symbol, $x = x_n x_{n-1}\cdots x_1$, $y = y_1 y_2 \cdots y_m$, and $z=z_1z_2\cdots z_l$ are sequences of n, m, and l letters in $\{a,b,c\}$, respectively, and $y', z' \in \{a,b,c\}^*$. Let i be an index such that $x_1x_2\cdots x_{i-1} = y_1y_2\cdots y_{i-1}$ and $x_i \neq y_i$. Let j be an index such that $x_1x_2\cdots x_{j-1} = z_1z_2\cdots z_{j-1}$ and $x_j \neq z_j$. It is promised that $y_i, z_j \neq a$ and ω satisfies either of the following two:

(p1) $|y'| = |y_{i+1}y_{i+2}\cdots y_m| = m-i$, $|z'| = |z_{j+1}z_{j+2}\cdots z_l| = l-j$, and $i=j$;
(p2) $|y'| \neq m-i$ and $|z'| \neq l-j$.

[Output] Decide whether the input satisfies (p1) and $y_i = z_j$. In that case the automaton *accepts* the input. If (p1) and $y_i \neq z_j$, or (p2) is satisfied, the automaton *rejects* it.

Problem I is a promise problem and we use the following setting : we decompose the set of input strings into "acceptable," "rejectable," and "don't care" inputs, and we identify only the "acceptable" and "rejectable" inputs correctly.

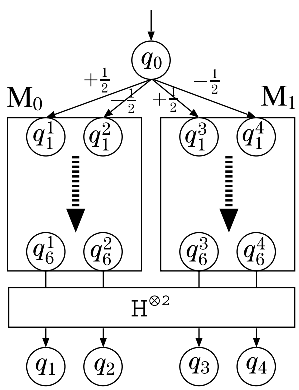


Fig. 1 QPA that solves Problem I deterministically.

Theorem 3.1. *There exists a QPA that solves Problem I deterministically.*

Proof. We construct a QPA $M = (Q,\Sigma,T,q_0, Q_{acc}, Q_{rej}, D, \delta)$ that solves Problem I deterministically as follows. $Q = Q_\downarrow \cup Q_\rightarrow$, where $Q_\downarrow = \{q_0, q_i, q^i_{rej}\}$ and $Q_\rightarrow = \{q^i_j\}$ $(1 \leq i \leq 4, 1 \leq j \leq 6)$, $\Sigma = \{a,b,c,\%\}$, $T = \{a,b,c,u\}$, $Q_{acc} = \{q_2\}$, $Q_{rej} = \{q_4, q^i_{rej}\}$, $D(q) =$ '$\rightarrow$' if $q \in Q_\rightarrow$, otherwise '$\downarrow$'. Transition function δ is defined as Fig. 2. Our main idea utilizes the Deutsch-Jozsa algorithm [11]. We sketch out the transition along with the algorithm:

$$|0\rangle|1\rangle$$
$$\xrightarrow{H^{\otimes 2}} \frac{1}{2}\{|M_0\rangle(|0\rangle-|1\rangle)+|M_1\rangle(|0\rangle-|1\rangle)\} \quad (1)$$
$$\xrightarrow{U_f} \frac{1}{2}\{|M_0\rangle(|0\oplus f(0)\rangle-|1\oplus f(0)\rangle)+ |M_1\rangle(|0\oplus f(1)\rangle-|1\oplus f(1)\rangle)\}, \quad (2)$$
$$= \frac{1}{2}(-1)^{f(0)}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle) \otimes(|0\rangle-|1\rangle), \quad (3)$$
$$\xrightarrow{H^{\otimes 2}} (-1)^{f(0)}|(f(0)\oplus f(1))\rangle|1\rangle, \quad (4)$$
$$= \begin{cases}(-1)^{f(0)}|0\rangle|1\rangle & \text{if } f \text{ is constant,}\\ (-1)^{f(0)}|1\rangle|1\rangle & \text{if } f \text{ is balanced.}\end{cases} \quad (5)$$

Let M_0 and M_1 represent 0 and 1, respectively, and $U_f : |x\rangle|y\rangle \longrightarrow |x\rangle|y\oplus f(x)\rangle$, where $f(0) = g(y_i)$, $f(1) = g(z_j)$, $g(b)=0$, and $g(c)=1$.

QPA M consists of two independent sub-QPAs, M_0 and M_1 (cf. **Fig. 1**), which have analogous behaviors. After reading the left endmarker, M goes to the superposed state of q^1_1, q^2_1, q^3_1, and q^4_1 with amplitudes $+\frac{1}{2}, -\frac{1}{2}, +\frac{1}{2}$, and $-\frac{1}{2}$, respectively. Expression (1) is considered to be this transition, e.g., $|M_0, 0\rangle$ represents state q_1^1 (to be exact, the configuration at q_1^1 containing the stack information and the position of the input tape head). M_0 is a subautomaton that starts in the superposition of q_1^1 and q_1^2, searches for i such that y_i first discords from x_i, and examines whether $|y_{i+1}\cdots y_m| = |y'|$. M_1 is also a subautomaton that starts in the superposition of q_1^3 and q_1^4, searches for j such that z_j first discords from x_j, and examines whether $|z_{j+1}\cdots z_l| = |z'|$. M_0 and M_1 run simultaneously. As will hereinafter be described in detail, M_0 and M_1 go to states $q_6^1, \ldots, q_6^4$ at the same time iff $i = j$, $|y_{i+1}\cdots y_m| = |y'|$, and $|z_{i+1}\cdots z_l| = |z'|$. Note that if $y_i(z_j)$ is b, the amplitudes of q_6^1 and q_6^2 (q_6^3 and q_6^4) are $+\frac{1}{2}$ and $-\frac{1}{2}$, while if $y_i(z_j)$ is c, then $-\frac{1}{2}$ and $+\frac{1}{2}$. These transitions correspond to Exp. (2), that is, the application of U_f denotes the simultaneous running of M_0 and M_1. For example, suppose that $i = j$, $y_i = b$, and $z_j = c$, the configuration of M

$$\frac{1}{2}\{(|q_6^1\rangle - |q_6^2\rangle) + (-|q_6^3\rangle + |q_6^4\rangle)\},$$

corresponds to Exp. (2)

$$\frac{1}{2}\{|M_0\rangle(|0\rangle - |1\rangle) + |M_1\rangle(|1\rangle - |0\rangle)\}. \quad (6)$$

By applying the Hadamard transform to Exp. (6), $|1\rangle|1\rangle$ is obtained, corresponding to q_4, namely, a rejecting state.

Note that this algorithm successfully functions iff condition (p1) is satisfied, since the two sub-QPAs must be in the superposed state of four q_6^i's at the same time and with the same stack configuration so that the interference of the second Hadamard transform is performed well. Thus, M can properly handle inputs that satisfy (p1). Before considering case (p2), we illustrate the sub-QPAs (cf. **Fig. 2**).

Since they have analogous behaviors as previously described, we will explain only one of them, M_0. Sub-QPA M_0

(1) reads x and puts it into the stack, remaining at q_1^1 and q_1^2;
(2) reads % and goes to the superposed state of q_2^1 and q_2^2;
(3) keeps retrieving a stack top symbol one by one at the superposed state until discordance between the stack top symbol and the input letter occurs, namely, y_i is read;
(4) reads y_i and pushes u into the stack, and goes to
  (a) q_3^1 from q_2^1 and q_3^2 from q_2^2 if $y_i = b$,
  (b) q_3^1 from q_2^2 and q_3^2 from q_2^1 if $y_i = c$;
(5) continues pushing a into the stack at the states while reading $y_{i+1}\cdots y_m$,
(6) reads %, goes to the superposed state of q_4^1 and q_4^2, and skips z at the state;
(7) reads %, goes to the superposed state of q_5^1 and q_5^2, and keeps retrieving a stack top one by one while reading y';
(8) reads %, goes to q_6^1 and q_6^2, and skips the remainder of the input.

Note that if the input satisfies (p1), M_0 and M_1 go to q_6^i's at the same time. Consider (p2). If $y_{i+1}\cdots y_m$ is shorter than y', at step (7) symbol u must show up at the stack top before reading through y' and M_0 goes to $q_{rej}^{1,2}$, namely, rejecting states. If $y_{i+1}\cdots y_m$ is longer, the stack top symbol will never be u when reading the right endmarker, and then the automaton goes to $q_{rej}^{1,2}$. Remember that M_1 has a similar behavior. Thus it is easy to show that the input satisfying (p2) leads both M_0 and M_1 to the rejecting states; disagreement of arrival timings have no need to be discussed. Therefore, M accepts input (p1) and rejects input (p2) with certainty.

Finally, we discuss the unitarity of the evolution of M. Obviously, the transition of M is reversible deterministic except for two Hadamard transforms. Thus, it is straightforward that the undefined transitions of δ can be defined properly to satisfy Well-formedness conditions. □

Further, we emphasize that this theorem also holds for 1 way QPAs. Our QPA can be seen as a 1 way QPA since the tape head always goes right except when it reads \$, or the finite state control comes to the accepting or rejecting state.

4. No DPAs Can Solve Problem I

In this section, we show that no DPAs can solve the problem defined in the previous section. We first present the generalized Ogden's lemma [14], which is one of the most useful results to give a proof that a language is not context-free.

Lemma 4.1. *For any context-free language L, $\exists n \in \mathbf{N}$ such that $\forall z \in L$, if p positions in z are "distinguished" and q positions are "excluded," with $p > n^{q+1}$, then $\exists u, v, w, x, y$, such that $z = uvwxy$ and;*

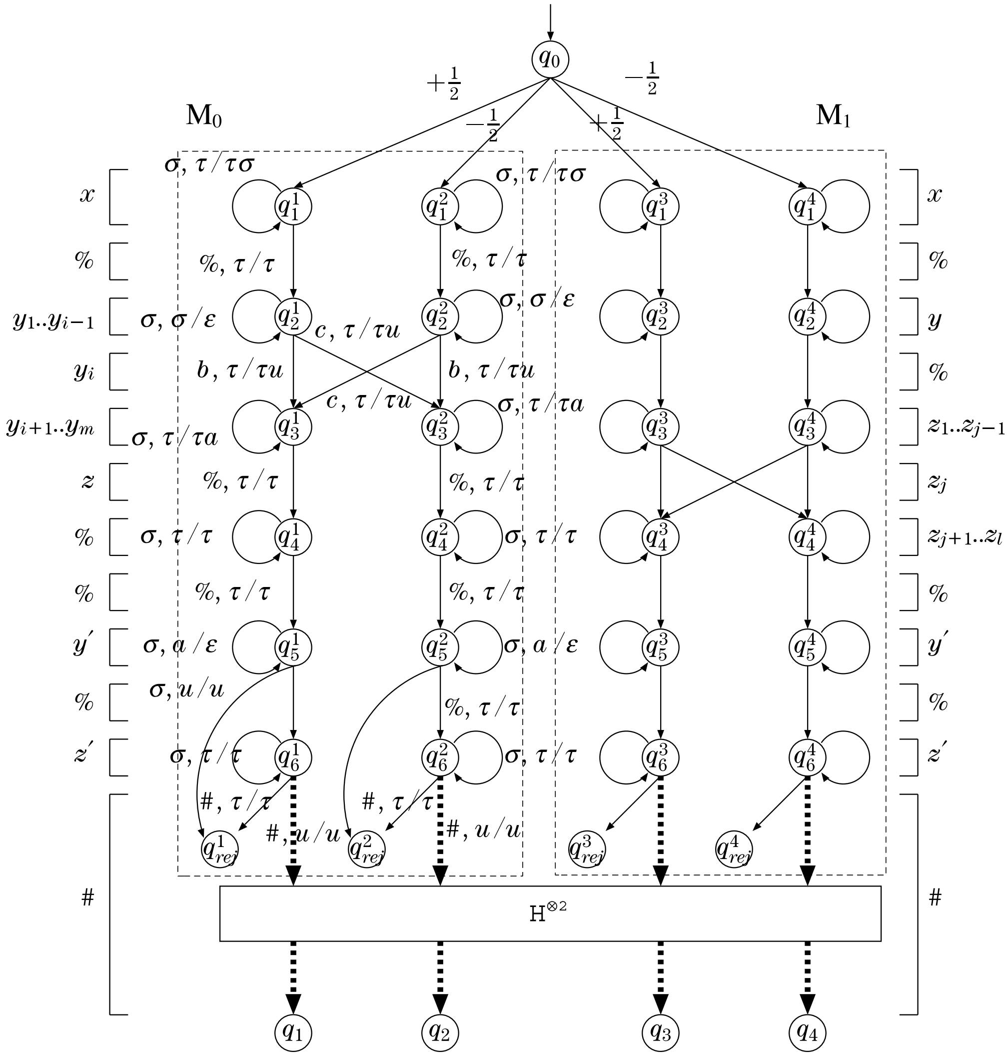


Fig. 2 The behaviors of the sub-QPAs. $(\sigma, \tau/\tau')$ represents the transition that when the input symbol is σ with the stack top τ, τ is retrieved and τ' is pushed into the stack, where $\sigma \in \Sigma$ and $\tau \in T$.

(1) *vx contains at least one distinguished position and no excluded positions;*

(2) *if p' is the number of distinguished positions and q' is the number of excluded positions in vwx, then $p' \le n^{q'+1}$;*

(3) *$\forall i \in \mathbf{N}$, $uv^iwx^iy \in L$.*

Proof. See the literature [14]. □

It is straightforward to see that the lemma can be applied not only to a string of terminal symbols but also to a string including non-terminal symbols or a string derived from a non-terminal symbol by a context-free grammar G. Thus, it is obvious that the following corollary holds.

Corollary 4.1. *For any context-free grammar G, $\exists n \in \mathbf{N}$ such that for all $z \in (T \cup V)^*$ derived from a non-terminal symbol (including a start symbol) X which is in itself derived by G, where T and V are sets of terminal and non-terminal symbols, respectively, if p positions in z are "distinguished" and q positions are "excluded," with $p > n^{q+1}$, then $\exists u, v, w, x, y$, such that $z = uvwxy$ and*

(1) *vx contains at least one distinguished position and no excluded positions,*

(2) *if p' is the number of distinguished positions and q' is the number of excluded*

positions in vwx, then $p' \leq n^{q'+1}$,

(3) $\forall i \in \mathbf{N}$, uv^iwx^iy is derived from X by G.

Since DPAs are special cases of non-deterministic pushdown automata, NPAs, the following theorem indicates that there are no DPAs that solve Problem I.

Theorem 4.1. *There exist no NPAs that solve Problem I.*

Proof. (Outline) If there were NPAs that solved Problem I, there would exist a context-free grammar G that derives every acceptable input string of the problem and some "don't care" strings, and does not derive any rejectable inputs. Thus, by Ogden's lemma, for any string z derived by G, there exists a decomposition $z = uvwxy$ such that for all $i \geq 0$, uv^iwx^iy is also derived by G. (cf. **Fig. 3**) We call such a decomposition *a good decomposition.* We will show that there exist no good decompositions, that is, G is not context-free.

However, Lemma 4.1 is insufficient for our purpose. Since Problem I is a promise problem, an awkward problem emerges that there can be a decomposition such that for some i, uv^iwx^iy is a "don't care" input derived by G. The modified Ogden's lemma, that is, Corollary 4.1 can be applied to the string to which the lemma or the corollary is already applied, so that such an awkward problem can be resolved as follows. If such an awkward decomposition is a good decomposition, there exists a non-terminal symbol X such that $uXy \overset{+}{\Rightarrow} uvXxy \overset{+}{\Rightarrow} uvwxy = z$, where '$A \overset{+}{\Rightarrow} B$' represents that A is derived from B by one or more applications of the production rule of G. For such a z, we consider $z' = uXy$ or $z' = w$. By Corollary 4.1, similarly, there exists a decomposition $z' = u'v'w'x'y'$ such that for all $j \geq 0$, $u'v'^jw'x'^jy'$ is also derived by G. (cf. **Figs. 4** and **5**) In this way, by implementing the independent multiparameter of iterations, say i and j such that $(uv^iwx^i..)v'^jw'x'^jy'$ in Fig. 4, we show the contradiction that for a certain string derived by G, there are no good decompositions.

(Details) Let L_1 be the set of YES instances of Problem I and L_2 be the set of NO instances, with $L_1 \cap L_2 = \phi$. We show that no NPAs can recognize any language that contains all $s \in L_1$ but does not contain any $s \in L_2$. Assume that there exists a context-free grammar G by

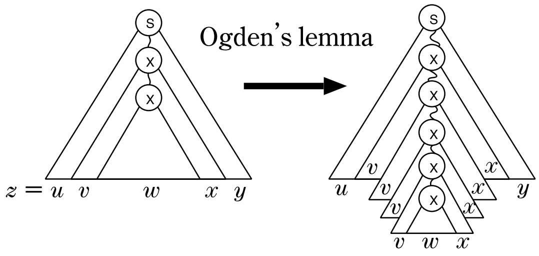


Fig. 3 Syntax trees of $z = uvwxy$ and uv^iwx^iy generated by G.

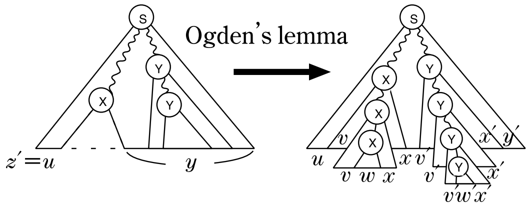


Fig. 4 Syntax trees of $z' = uXy$ and $(uv^iwx^i..)v'^jw'x'^jy'$ generated by G.

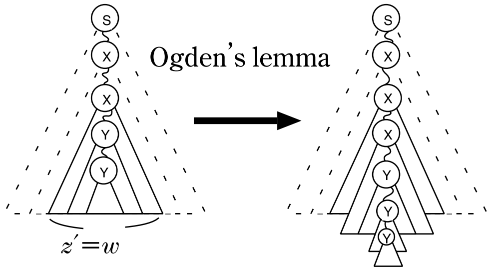


Fig. 5 Syntax trees of $z' = w$ and $u'v'^j(..v^iwx^i..)x'^jy'$ generated by G.

which all $s \in L_1$ and no $s \in L_2$ are derived. By Lemma 4.1, we can decompose $s \in L_1$, where $|s| > n$ and n is the constant of the lemma, as $s = uvwxy$ such that for all i, uv^iwx^iy is derived by G.

We consider a string $s_1 = ac_1^Nb_1^N\%b_2^Nc_2^N\hat{b}b_3^N\%b_4^Nc_3^N\hat{b}c_4^N\%b_5^N\%c_5^N \in L_1$, where b_i and $\hat{b}$ represent the letters 'b' and c_i does 'c'. Hereafter, throughout this proof, we let $a, \hat{b}$, %, and both end letters of b_i's and c_i's be excluded. Let the number of the excluded be $p\,(=27)$ and let $N = n^{p+1} + 3$. Let each letter of b_1's be distinguished except both end letters (which are excluded). By Lemma 4.1, $\exists u_1, v_1, w_1, x_1, y_1$ such that $s_1 = u_1v_1w_1x_1y_1$ and $\forall i \geq 0$, $u_1v_1^iw_1x_1^iy_1$ is derived by G. We consider the following three cases as candidates of good decompositions and show that none of them are good decomposi-

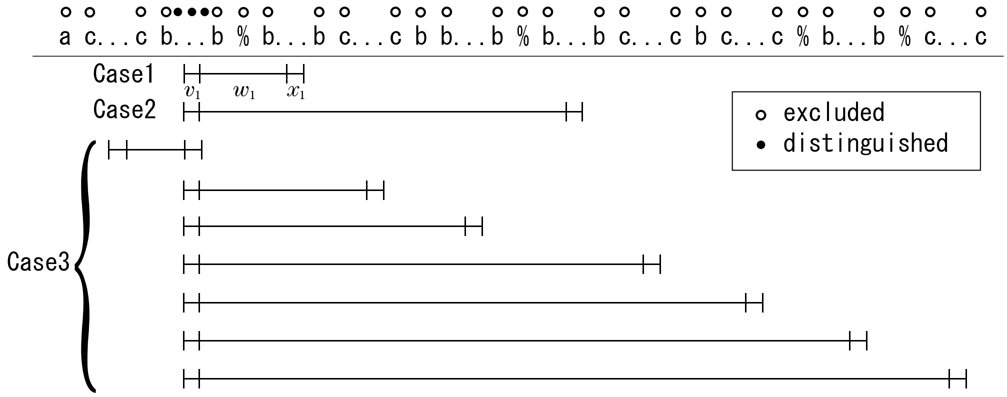


Fig. 6 Decompositions of Cases 1, 2, and 3.

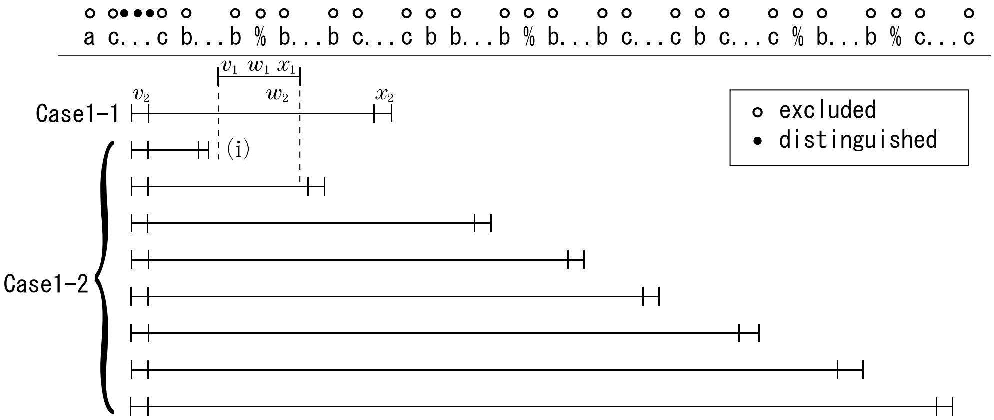


Fig. 7 Decompositions of Cases 1-1 and 1-2.

tions, leading to a contradiction.

Case 1: $v_1 = b_1^+$, $x_1 = b_2^+$, and $|v_1| = |x_1|$;
Case 2: $v_1 = b_1^+$, $x_1 = b_4^+$, and $|v_1| = |x_1|$;
Case 3: otherwise.

Figure 6 illustrates intuitively how each case decomposes s_1. Consider Case 1:

$$s_1 = \underbrace{ac_1^N b_1..}_{u_1}\ \underbrace{...}_{v_1}\ \underbrace{..b_1\%b_2..}_{w_1}\ \underbrace{...}_{x_1}\ \underbrace{..b_2c_2^N..c_5^N}_{y_1}.$$

Note that for all i, $u_1v_1^iw_1x_1^iy_1 \notin L_2$. Thus we consider the string $u_1X_1y_1$, where X_1 is a non-terminal symbol such that $u_1X_1y_1 \overset{+}{\Rightarrow} u_1v_1X_1x_1y_1 \overset{+}{\Rightarrow} u_1v_1w_1x_1y_1$. Let $s_2 = u_1X_1y_1$ and let each letter of c_1's except both end letters be distinguished. By Corollary 4.1, $\exists u_2, v_2, w_2, x_2, y_2$ such that $s_2 = u_2v_2w_2x_2y_2$ and $\forall j \geq 0$, $u_2v_2^jw_2x_2^jy_2$ is derived by G. We consider the following two cases as candidates of good decompositions (**Fig. 7**).

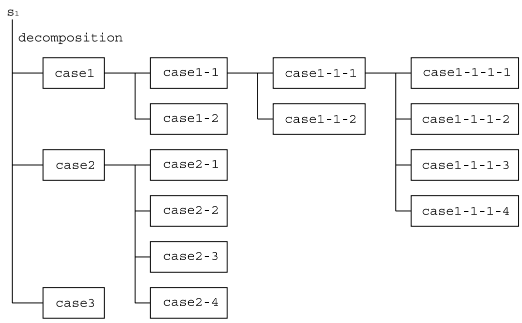


Fig. 8 Layered decomposition.

Case 1-1: $v_2 = c_1^+$, $x_2 = c_2^+$, and $|v_2| = |x_2|$;
Case 1-2: otherwise.

Afterward, in this way we employ a layered decomposition as shown in **Fig. 8**. If none of the lower layers are good decompositions, it is assured that the upper layer is not a good decom-

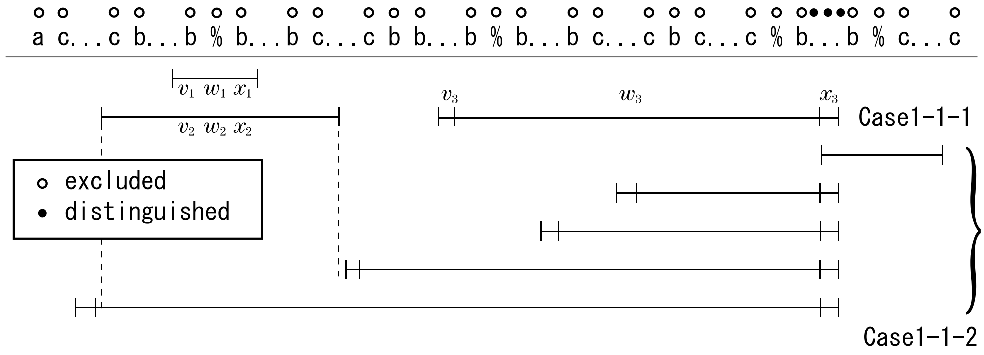


Fig. 9 Decompositions of Cases 1-1-1 and 1-1-2.

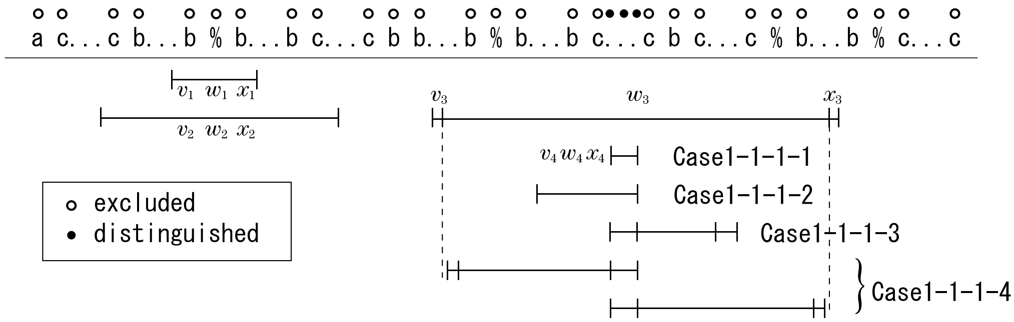


Fig. 10 Decompositions of each case.

position. Consider Case 1-2 ((i) in Fig. 7).

$$s_2 = \underbrace{ac_1..}_{u_2}\ \underbrace{...}_{v_2}\ \underbrace{..c_1b_1..}_{w_2}\ \underbrace{...}_{x_2}\ ..b_2c_2..\ \underbrace{X_1}_{v_1w_1x_1}\ ...c_5.$$

For $i = 1$ and $j = 0$, $u_2v_2^jw_2x_2^j(u_1'v_1^iw_1x_1^iy_1) = ac^{N-|v_2|}\ b^{N-|x_2|}\ \%b^N\ c^Nbb^N\ \%\ b^Nc^Nbc^N\ \%b^N\%c^N$, where the round brackets stand for the substring y_2. This string satisfies (p2) and so is in L_2. Thus, this is not a good decomposition. Similarly, all of the others in Case 1-2 are not good decompositions. Consider Case 1-1. Note that for all i and j, $u_2v_2^j(..u_1^iw_1x_1^i..)x_2^jy_2 \notin L_2$. Let X_2 be a non-terminal symbol such that $u_2X_2y_2 \overset{+}{\Rightarrow} u_2v_2X_2x_2y_2 \overset{+}{\Rightarrow} u_2v_2w_2x_2y_2$. Let $s_3 = u_2X_2y_2$ and let each letter of b_5's except both end letters be distinguished. By Corollary 4.1, $\exists u_3, v_3, w_3, x_3, y_3$ such that $s_3 = u_3v_3w_3x_3y_4$ and $\forall k \geq 0$, $u_3v_3^kw_3x_3^ky_3$ is derived by G. We consider the following two cases as candidates of good decompositions (**Fig. 9**).

Case 1-1-1: $v_3 = b_3^+$, $x_3 = b_5^+$, and $|v_3| = |x_3|$;
Case 1-1-2: otherwise.

In Case 1-1-2, it can be shown that there exist some i, j, and k such that respective decompositions are not good decompositions. Consider Case 1-1-1. Note that for all i, j and k, $(u_2v_2^j(..u_1^iw_1x_1^i..)x_2^j..)v_3^kw_3x_3^ky_3 \notin L_2$. Let X_3 be a non-terminal symbol such that $u_3X_3y_3 \overset{+}{\Rightarrow} u_3v_3X_3x_3y_3 \overset{+}{\Rightarrow} u_3v_3w_3x_3y_3$. Let $s_4 = w_3$ and let each letter of c_3's except both end letters be distinguished. By Corollary 4.1, $\exists u_4, v_4, w_4, x_4, y_4$ such that $s_4 = u_4v_4w_4x_4y_4$ and $\forall l \geq 0$, $u_4v_4^lw_4x_4^ly_4$ is derived from X_3 by G. We consider the following four cases as candidates of good decompositions (**Fig. 10**).

Case 1-1-1-1: $v_4x_4 = c_3^+$;
Case 1-1-1-2: $v_4 = b_4^+$, $x_4 = c_3^+$;
Case 1-1-1-3: $v_4 = c_3^+$, $x_4 = c_4^+$;
Case 1-1-1-4: otherwise.

Consider Case 1-1-1-1. Note that for all i, j, k and l, $(u_2v_2^j(..u_1^iw_1x_1^i..)x_2^j..)v_3^k(..u_4^lw_4x_4^l..)x_3^ky_3 \notin L_2$. Let X_5 be a non-terminal symbol such that $u_5X_5y_5 \overset{+}{\Rightarrow} u_5v_5X_5x_5y_5 \overset{+}{\Rightarrow} u_5v_5w_5x_5y_5$. Let $s_5 = u_3X_3y_3$ and let each letter of c_5's except both end letters be distinguished. By Corollary 4.1, $\exists u_5, v_5, w_5, x_5, y_5$ such that $s_5 = u_5v_5w_5x_5y_5$ and $\forall m \geq 0$, $u_5v_5^mw_5x_5^my_5$ is derived by G. We consider the following five cases

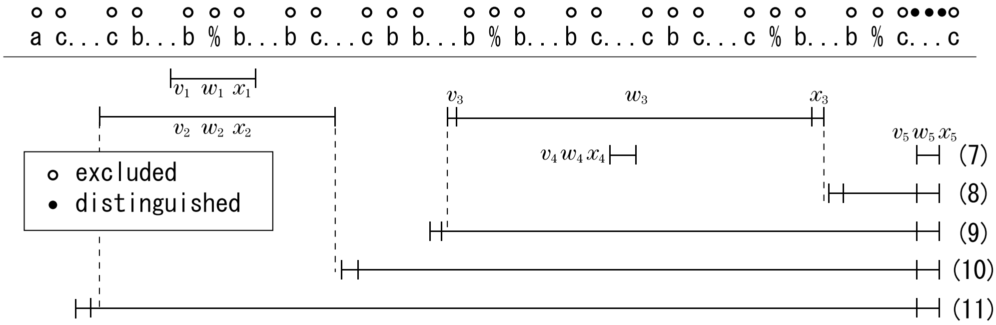


Fig. 11 Decompositions of (7)..(11).

(**Fig. 11**).

$$v_5 x_5 = c_5^+, \tag{7}$$
$$v_5 = b_5^+ \textit{ and } x_5 = c_5^+, \tag{8}$$
$$v_5 = b_3^+ \textit{ and } x_5 = c_5^+, \tag{9}$$
$$v_5 = c_2^+ \textit{ and } x_5 = c_5^+, \textit{and} \tag{10}$$
$$v_5 = c_1^+ \textit{ and } x_5 = c_5^+. \tag{11}$$

As shown below, for each of the above there exist i, j, k, l, and m such that the iterated string is in L_2.

In case (7), for $i = j = k = 1$, $(l-1)|v_4x_4| = (m-1)|v_5x_5|$, $ac^Nb^N\%b^Nc^Nbb^N\%b^Nc^{N_1}bc^N\%b^N\%c^{N_2} \in L_2$, where $N_1 = N+(l-1)|v_4x_4|$ and $N_2 = N+(m-1)|v_5x_5|$.

In case (8), for $i = j = k = 1$, $l = m = 0$, $ac^Nb^N\%b^Nc^Nbb^N\%b^Nc^{N_1}bc^N\%b^{N_2}\%c^{N_3} \in L_2$, where $N_1 = N - |v_4x_4|$, $N_2 = N - |v_5|$ and $N_3 = N - |x_5|$.

In case (9), for $i = j = k = 1$, $l = m = 0$, $ac^Nb^N\%b^Nc^Nbb^{N_1}\%b^Nc^{N_2}bc^N\%b^N\%c^{N_3} \in L_2$, where $N_1 = N - |v_5|$, $N_2 = N - |v_4x_4|$ and $N_3 = N - |x_5|$.

In case (10), for $i = j = k = l = 1$, and $m = 2$, $ac^Nb^N\%b^Nc^{N_1}bb^N\%b^Nc^Nbc^N\%b^N\%c^{N_2} \in L_2$, where $N_1 = N + |v_5|$ and $N_2 = N + |v_4x_4|$.

In case (11), for $i = j = k = l = 1$, and $m = 0$, $ac^Nb^N\%b^Nc^{N_1}bb^N\%b^Nc^Nbc^N\%b^N\%c^{N_2} \in L_2$, where $N_1 = N - |v_5|$ and $N_2 = N - |v_4x_4|$.

The same goes for Cases 1-1-1-2, 1-1-1-3, and 1-1-1-4. Thus, Case 1 is not a good decomposition. Cases 2 and 3 are also similar to Case 1. Therefore, there exist no good decompositions on $s_1 \in L_1$. □

5. Conclusions and Future Works

In the third section, we showed that QPAs can solve a certain problem deterministically. The inputs of the problem are strings in the form of $x\%y\%z\%y'\%z'$. To construct such QPAs, we utilized two sub-QPAs, where one examined some relationships among x and y and y', and the other examined some relationships among x and z and z'. We ran the two sub-QPAs in parallel and utilized the Deutsch-Jozsa algorithm, which is a deterministic quantum algorithm for Deutsch's XOR problem, when we got a deterministic solution.

Furthermore, in the fourth section, we showed that no DPAs can solve the problem by using extended generalized Ogden's lemma.

We should consider languages recognized by QPAs but not by DPAs, as future work.

Acknowledgments This work was supported in part by Grants-in-Aid for Scientific Research (No.15700014 and No.16092218).

References

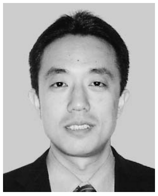
1) Kondacs, A. and Watrous, J.: On the power of quantum finite state automata, *Proc. FOCS'97*, pp.66–75 (1997).
2) Ambainis, A. and Freivalds, R.: 1-way quantum finite automata: strengths, weaknesses and generalizations, *Proc. FOCS'98*, pp.332–341 (1998).
3) Kravtsev, M.: Quantum Finite One-Counter Automata, *Proc. SOFSEM 1999*, Vol.1725 of Lecture Notes in Computer Science, pp.431–441 (1999).
4) Bonner, R.F., Freivalds, R. and Kravtsev, M.: Quantum versus probabilistic one-way finite automata with counter, *Proc. SOFSEM 2001*, Vol.2234 of Lecture Notes in Computer Science, pp.181–191 (2001).
5) Yamasaki, T., Kobayashi, H., Tokunaga, Y. and Imai, H.: One-way probabilistic reversible and quantum one-counter automata, *Theoretical Computer Science*, Vol.289, No.2, pp.963–

976 (2002).
6) Yamasaki, T., Kobayashi, H. and Imai, H.: Quantum versus deterministic counter automata, Proc. COCOON 2002, LNCS 2387, pp.584–594 (2002).
7) Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput., Vol.26, 1484–1509 (1997).
8) Grover, L.: A fast quantum mechanical algorithm for database search, *Proc. STOC'96*, pp.212–219 (1996).
9) Moore, C. and Crutchfield, J.P.: Quantum Automata and Quantum Grammars, *Theoretical Computer Science*, Vol.237, No.1-2, pp.275–306 (2000).
10) Golovkins, M.: Quantum Pushdown Automata, *SOFSEM 2000*, LNCS 1963, pp.336–346 (2000).
11) Deutsch, D. and Jozsa, R.: Rapid Solution of Problem by Quantum Computation, *Proc. R. Soc. Lond.*, A Vol.439, pp.553–558 (1992).
12) Cleve, R., Ekert, A.K., Macchiavello, C., and Mosca, M.: Quantum algorithms revisited, *Proc. R. Soc. Lond.*, A Vol.454, pp.339–354 (1998).
13) Simon, D.R.: On the power of quantum computation, *SIAM J. Comput.*, Vol.26, No.5, pp.1474–1483 (1997).
14) Bader, C. and Moura, A.: A Generalization of Ogden's lemma, Journal of the Association for Computing Machinery, Vol.29, No.2, pp.404–407 (1982).
15) Amano, M., Iwama, K. and Raymond, R.: Exploiting the Difference in Probability Calculation between Quantum and Probabilistic Computations, IEICE Trans. Fundamentals, Vol.E-87A, No.5, pp.1004–1011 (2004).
16) Buhrman, H., Cleve, R. and Wigderson, A.: Quantum vs. classical communication and computation, *Proc. 30th Annual ACM Symposium on Theory of Computing*, pp.63–68 (1998).
17) Raz, R.: Exponential separation of quantum and classical communication complexity, *Proc. 31st Annual ACM Symposium on Theory of Computing*, pp.358–367 (1999).

(Received January 31, 2005)
(Accepted July 4, 2005)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.1, pp.426–435.)

Yumiko Murakami was born in 1978. She received the B.E. and M.E. degrees from Tokyo Institute of Technology and Nara Institute of Science and Technology, respectively. She is currently working toward the Ph.D. degree at Graduate School of Information Science, Nara Institute of Science and Technology. Her currenet interests include quantum computation and cryptographic protocols.

Masaki Nakanishi was born in Osaka, Japan, in 1973. He received the B.E., M.E. and Ph.D. degrees from Osaka University, Japan, in 1996, 1998 and 2002 respectively. He is currently with the Graduate School of Information Science, Nara Institute of Science and Technology, as a Research Associate. His current interests include quantum computation and design of combinatorial algorithms.



Shigeru Yamashita received the B.E., M.E. and Ph.D. degrees in Information Sience from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2001, respectively. He is a Associate Professor of Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include new types of computation including quantum computation and reconfigurable computing. He received the 2000 IEEE Circuits and Systems Society Transactions on Computer-Aided Design of Integrated Circuits and Systems Best Paper Award.

Katsumasa Watanabe received the B.E., M.E. and Dr. Eng. degrees from Kyoto University, Kyoto, Japan. He is a Professor at Nara Institute of Science and Technology from 1992. His current interests include the active software, and the design and implementation of programming languages in hardware/software codesign environment.