# Quantum Computation and Learning

Richard Bonner
Rūsiņš Freivalds
(Eds.)

Third International Workshop, QCL 2002
Riga, Latvia
Revised Proceedings

ISBN 91-88834-29-8
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Department of Mathematics and Physics,
Mälardalen University, Västerås, Sweden

# Preface

In the background of any model of computation lies an assumption of a physical computational device, a physical system, the controlled dynamics of which is the computation. Classical models of computation tacitly assume an underlying classical and usually rather simple physical system. However, the miniaturization of electronic circuits on the one hand, and fundamental scientific questions on the other, have in recent years put this tacit assumption into question, giving birth to a rapidly growing field of *quantum* computation and information theory. Information may thus be stored in quantum physical systems and processed by controlling their dynamics.

The present volume contributes to this new field. It is a collection of revised papers, originally presented at the international workshop QCL'2002 in Riga in May 2002. The papers are grouped into two separate sections, *Quantum Computation*, and, *Learning*, reflecting the fact that Quantum Learning - a goal of our investigations - is still a field in the making. Let us very briefly review the topics covered.

The simplest model of quantum computation is the quantum finite automaton (QFA). There are several versions of this model, differentiated essentially by the mode of data access (1-way, 2-way, or hybrid), the way the quantum states are measured (measure-once or measure-many), the kind of quantum states admitted (pure or mixed), and the criteria of input acceptance (isolated or non-isolated cut-point).

Five of the presented papers fall into the category of general QFA theory, which compares different QFA with one another, and with their classical, deterministic or probabilistic, counterparts. Golovkins and Kravtsev propose a new class of probabilistic automata with doubly stochastic transition matrices; this class is close, in a well-defined sense, to the QFA. Bonner, Freivalds and Rasščevskis find a language for the recognition of which the number of states required by a 1-way measure-many QFA is exponentially smaller than that required by a classical deterministic automaton; Midrijanis, on the other hand, presents another language, for which the relation is roughly the opposite. Ozols finds a language not recognized by deterministic automata, but recognized by 1-way measure-many QFA with non-isolated cut-point. Dubrovsky and Scegulnaja compare QFA with pure states and QFA with mixed states with respect to Boolean function computation and language recognition; they present a language recognized by the QFA with mixed states with a better probability than achievable by admitting pure states only.

Four authors consider QFA in a more specific context: V. Kravcevs, Lace and Kuzmenko - for Boolean function computation, and Tervits - for undecidability proofs. Kravcevs and Lace compare the performance of quantum and probabilistic query automata (decision trees) on specific functions, Kuzmenko constructs quantum automata for the computation of certain DNF forms in a single query,

and Tervits constructs a QFA which recognizes a language of key importance to an indecidability problem.

Two authors, Kuzmenko (*Query Automata Minimization*) and Luchko, consider the problem of evolving a quantum automaton for a given task by means of genetic algorithms.

There are three papers on classical learning. Tervits considers finite standardizability, proving that this type of learning is not much less powerful than Gold's identification in the limit. Kaulis studies the synthesis of logic formulae from finite examples, obtaining necessary and sufficient conditions in many-sorted first-order predicate logic with equality. Greizina and Grundmane take up a special problem of *Learning from Zero Information*.

Finally, quantum learning is considered in the two papers by Bonner and Freivalds. One of the papers discusses limited memory learning by 1-way QFA in the 'identification in the limit' model, showing that it is more powerful than its classical counterpart. The other paper reviews recent results on quantum learning.

We thank all contributors, referees, and our Program Committee. We gratefully acknowledge financial sponsorship from several institutions, specified below and in the papers, and in particular the initial support of the Swedish Institute for our ML2000 project, which stared it all.

Richard Bonner
Rūsiņš Freivalds

# Organization

The International Workshop QCL'2002, on Quantum Computation and Learning, was organized jointly by the Institute of Mathematics and Computer Science, University of Latvia, Riga, and the Department of Mathematics and Physics, Mälardalen University, Västerås, Sweden. It was the third of annual QCL workshops, started within a joint research project ML2000, under financial sponsorship of the Swedish Institute. Working papers from the previous workshops, QCL'1999 and QCL'2000, are available online at
http://www.ima.mdh.se/personal/rbr/courses/QCL.htm.

## Program Committee

Farid Ablayev (Kazan, Russia)
Richard Bonner (Västerås, Sweden)
Cristian S. Calude (Auckland, New Zealand)
Alexandre Dikovsky (Nantes, France)
Rusins Freivalds (Riga, Latvia)
Alexander Letichevsky (Kiiv, Ukraine)
Samuel J. Lomonaco, Jr. (Baltimore, U.S.A.)
Arun Sharma (Sydney, Australia)
Mars Valiev (Moscow, Russia)

## Research and Workshop Sponsors

The European Commission
The Latvian Council of Science
The Swedish Institute
A/S Dati Grupa
The University of Latvia
The Department of Mathematics and Physics, Mälardalen University

# Table of Contents

## Quantum Computation



## Learning

# Probabilistic Reversibility and its Relation to Quantum Automata

Marats Golovkins * and Maksim Kravtsev**

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
marats@latnet.lv, maksims@batsoft.lv

**Abstract.** To study the relationship between quantum finite automata and probabilistic finite automata, we introduce a notion of probabilistic reversible automata (PRA), or doubly stochastic automata. We find that there is a strong relationship between different possible models of PRA and corresponding models of quantum finite automata. We also propose a classification of reversible finite 1-way automata.

## 1 Introduction

Two models of probabilistic reversible automata (PRA) are examined in this paper, namely, 1-way PRA and 1.5-way PRA. Presently, we outline the notions applicable to both models in a quasi-formal way, including a general notion of probabilistic reversibility; formal definitions are provided in further sections.

*Introductory notions* If not specified otherwise, we denote by $\Sigma$ an input alphabet of an automaton, $\Sigma^*$ then being the set of its finite words. Every input word in $\Sigma^*$ is assumed enclosed within *end-marker* symbols # and \$; it is therefore convenient to also introduce a *working alphabet* $\Gamma = \Sigma \cup \{\#, \$\}$. By $Q$ we normally understand the set of states of an automaton. By $\overline{L}$ we understand the complement of a language $L \subset \Sigma^*$. For an input word $\omega$ in $\#\Sigma^*\$$, we denote by $|\omega|$ the number of symbols in $\omega$, and by $[\omega]_i$, $i = 1, 2, \ldots$, the $i$-th consecutive symbol of $\omega$, excluding end-markers.

Presented with a word $\omega$ on its input tape, a probabilistic automaton reads $\omega$ one symbol at a time, each time changing state; the next symbol and next state being determined only in probabilistic sense. In particular, we write $q \xrightarrow{S} q'$, $S \subset \Sigma^*$, if there is a positive probability of getting from state $q$ to state $q'$ by reading $\omega \in S$. We define the *configuration* of an automaton as a triple $c = \langle \nu q \nu' \rangle$ with $q \in Q$ and $\nu\nu' \in \#\Sigma^*\$$; it is understood that the input tape head of the automaton is above the first symbol in the word $\nu'$. To make accessible

* Research partially supported by the Latvian Council of Science, grant No. 01.0354 and grant for PhD students; University of Latvia, K. Morbergs grant; European Commission, contract IST-1999-11234

** Research partially supported by the Latvian Council of Science, grant No. 01.0354 and European Commission, contract IST-1999-11234

configurations of type $\langle q\#\omega\$\rangle$, $q \in Q$, $\omega \in \Sigma^*$, we assume that each word is written on a circular tape, and that the next symbol after the right end-marker \$ is the left end-marker #; this precondition is also used for quantum finite automata, see, for example, [KW 97].

Clearly, with a finite word $\omega \in \#\Sigma^*\$$ on its input tape, a finite automaton can access a set $C_\omega$ of configurations of at most $|\omega||Q|$ elements. The set $C = \bigcup_\omega C_\omega$ of configurations with arbitrary input is, on the other hand, countably infinite. A probabilistic automaton in configuration $c \in C_\omega$ chooses its next configuration according to a probability distribution $p_c$ on $C_\omega$, called a *superposition* of configurations; in case of 1-way automata, for example, the letter to be read next is determined, so superpositions involve states in $Q$ only. The set $\mathbb{R}^{C_\omega}$ of all real-valued functions on $C_\omega$ is a real vector space, where elements of $C_\omega$ can be viewed as basis vectors; this basis is called the *canonical basis*. A probabilistic automaton determines a linear endomorphism $A_\omega$ of $\mathbb{R}^{C_\omega}$, defined by $A_\omega(c) = p_c$ for basis vectors $c \in C_\omega$; if defined consistently, the operators $A_\omega$, $\omega \in \Sigma^*$, extend to a linear endomorphism $A$ of $\mathbb{R}^C$, called the *transition-* or the *evolution* operator of the automaton.

*Probabilistic reversible automata* Consider Nayak's model [N 99] of quantum automata with mixed states, where evolution is characterized by a unitary matrix, measurements are performed after each step, and POVM measurements not allowed. If the result of every measurement is a single configuration, not a proper superposition, we actually get a probabilistic automaton. Moreover, the evolution matrices will then be *doubly* stochastic. This motivates us to call a probabilistic finite automaton *reversible* if the matrix of its transition operator in the canonical basis is doubly stochastic.

*Word acceptance and language recognition* We admit two natural notions of word acceptance for reversible automata, speaking henceforth of C-automata and DH-automata when referring to these notions:

**Definition 1.** Classical acceptance. *An automaton accepts (rejects) a word classically, if its set of states consists of two disjoint subsets, of accepting and rejecting states, and the automaton enters an accepting (rejecting) state upon reading the last symbol of the word.*

**Definition 2.** Decide-and-halt acceptance. *An automaton accepts (rejects) a word $\omega$ in a decide-and-halt manner, if its set of states consists of three disjoint subsets, of accepting, rejecting, and non-halting states, and, when processing $\omega$, the automaton halts as soon as it enters an accepting (rejecting) state.*

We consider only bounded error language recognition; we define it in an equivalent way to that in [R 63].

**Definition 3.** *Let $A$ be an automaton with alphabet $\Sigma$, $L \subset \Sigma^*$ - a language, and let $P_\omega$ denote the probability that a word $\omega \in \Sigma^*$ is accepted by $A$. We say,*

1. *an automaton $A$ recognizes $L$ with bounded error and interval $(p_1, p_2)$ if* $0 \leq p_1 = \sup\{P_\omega \mid \omega \notin L\} < p_2 = \inf\{P_\omega \mid \omega \in L\}$,
2. *an automaton $A$ recognizes $L$ with probability $p$ ($p > \frac{1}{2}$) if $A$ recognizes $L$ with interval $(1-p, p)$,*
3. *a class $\mathcal{A}$ of automata recognizes $L$ with probability $1-\varepsilon$, if for every $\varepsilon > 0$ there exists an automaton $A \in \mathcal{A}$, which recognizes $L$ with interval $(\varepsilon_1, 1-\varepsilon_2)$, where $\varepsilon_1, \varepsilon_2 \leq \varepsilon$.*

*Quantum finite automata* We refer to several existing models of quantum finite automata (QFA): the measure-once [MC 97] (QFA-MC), the measure-many [KW 97] (QFA-KW), and the enhanced [N 99] (QFA-N) model. We note that QFA-MC are C-automata whereas QFA-KW and QFA-N are DH-automata.

*Results* Our results are presented in four subsequent sections. In Section 2, we discuss properties of PRA C-automata (PRA-C). We show that the class of languages recognized by PRA-C is closed under boolean operations, inverse homomorphisms and word quotient, but it is not closed under homomorphisms. We prove that PRA-C recognize the class of languages $a_1^* a_2^* \ldots a_n^*$ with probability $1-\varepsilon$; this class can be recognized by QFA-KW but with worse acceptance probabilities [ABFK 99]. It follows that QFA-N recognize this class of languages with probability $1-\varepsilon$. Further, we exhibit a general class of regular languages not recognizable by PRA-C, which in particular contains the languages $(a,b)^*a$ and $a(a,b)^*$; this class has strong similarities with the class of languages not recognizable by QFA-KW [AKV 00]. In Section 3 we prove that PRA DH-automata do not recognize the language $(a,b)^*a$. In Section 4 we discuss some properties of 1.5-way PRA and present an alternative notion of probabilistic reversibility, not connected with quantum automata. In the last Section 5 we propose a classification of reversible automata: deterministic, probabilistic and quantum.

Finally, the Appendix contains background material on doubly stochastic finite Markov chains, and a treatment of probabilistic reversible automata without end-markers. We show there that the use of end-markers does not affect the computational power of PRA-C: for every PRA-C with end-markers recognizing a language there is a PRA-C without end-markers recognizing the same language.

## 2 1-way probabilistic reversible C-automata

**Definition 4.** [1-way PRA-C] *A 1-way finite **probabilistic automaton** (PA) is specified by a finite set of states $Q$, a finite input alphabet $\Sigma$, and a transition function*

$$\delta : Q \times \Gamma \times Q \longrightarrow \mathbb{R}_{[0,1]},$$

*where $\Gamma = \Sigma \cup \{\#, \$\}$ is the input tape alphabet of $A$ and $\#$, $\$$ are end-markers not in $\Sigma$; for all $q \in Q$ and $\sigma \in \Gamma$, the transition function is required to satisfy:*

$$\sum_{q' \in Q} \delta(q, \sigma, q') = 1. \tag{1}$$

*A 1-way finite probabilistic automaton is a probabilistic* ***reversible*** *automaton (PRA) if for all $q \in Q$ and $\sigma \in \Gamma$,*

$$\sum_{q' \in Q} \delta(q', \sigma, q) = 1. \tag{2}$$

*A PRA-C is a PRA with classical word acceptance (Definition 1): there is an initial state $q_0 \in Q$, in which the processing of all words begins, and a set of accepting states $Q_F \subseteq Q$, which determine whether the word is accepted upon completed processing of the word's last letter. All in all, a PRA-C is specified by a tuple $A = (Q, \Sigma, q_0, Q_F, \delta)$.*

Note that for every input symbol $\sigma \in \Gamma$, the transition function of a 1-way finite PA is determined by a $|Q| \times |Q|$ matrix $V_\sigma$ with $(V_\sigma)_{ij} = \delta(q_j, \sigma, q_i)$, assuming a numbering of states. The conditions (1) and (2) then say that the matrices $V_\sigma$, $\sigma \in \Gamma$, are column- and row-stochastic, respectively; if both conditions hold, the matrices are doubly stochastic. We associate with a 1-way finite PA a linear evolution operator $A$ in $\mathbb{R}^C$ by putting

$$Ac = \sum_{q' \in Q} \delta(q, \sigma, q') \langle \nu \sigma q' \nu' \rangle.$$

for $c = \langle \nu q \sigma \nu' \rangle \in C$, and extending to $\mathbb{R}^C$ by linearity. It is immediate by (1) and (2) that the infinite matrix of $A$ in the canonical basis $C$ is doubly stochastic. This completes our formal definition of PRA-C.

### 2.1 Probability boosting

We consider language recognition by PRA-C in the sense of Definition 3.

**Theorem 1.** *If a language is recognized by a PRA-C, it is recognized by PRA-C with probability $1 - \varepsilon$.*

*Proof.* We follow the standard majority-voting argument. Let $L$ be a language recognized by PRA-C $A = (Q, \Sigma, q_0, Q_F, \delta)$ with interval $(p_1, p_2)$. Put $\bar{p} = \frac{1}{2}(p_1 + p_2)$, and let $A_m$ be a system of $m$ copies of $A$ 'working in parallel', which accepts a word when more than $m\bar{p}$ automata in the system have accepted the word, and otherwise rejects the word.

Take $\omega \in L$. The automaton $A$ accepts $\omega$ with probability $p_\omega \geq p_2$. As a result of reading $\omega$, $\mu_m^\omega$ automata of the system accept the word, and the rest reject it. The system has accepted the word, if $\frac{\mu_m^\omega}{m} > \bar{p}$. Pick $\eta_0$ so that $0 < \eta_0 < p_2 - \bar{p} \leq p_w - \bar{p}$. Estimate the probability that $\frac{\mu_m^\omega}{m} > \bar{p}$:

$$P\left\{\frac{\mu_m^\omega}{m} > \bar{p}\right\} \geq P\left\{p_\omega - \eta_0 < \frac{\mu_m^\omega}{m} < p_\omega + \eta_0\right\} = P\left\{\left|\frac{\mu_m^\omega}{m} - p_\omega\right| < \eta_0\right\}. \tag{3}$$

In case of $m$ Bernoulli trials, Chebyshev's inequality yields ([GS 97], p. 312):

$$P\left\{\left|\frac{\mu_m^\omega}{m} - p_\omega\right| \geq \eta_0\right\} \leq \frac{p_\omega(1 - p_\omega)}{m\eta_0^2} \leq \frac{1}{4m\eta_0^2}, \tag{4}$$

which implies

$$P\left\{\left|\frac{\mu_m^\omega}{m} - p_\omega\right| < \eta_0\right\} \geq 1 - \frac{1}{4m\eta_0^2}. \tag{5}$$

By (3) and (5),

$$P\left\{\frac{\mu_m^\omega}{m} > \bar{p}\right\} \geq 1 - \frac{1}{4m\eta_0^2}. \tag{6}$$

Note that (6) holds for every $\omega \in L$.

Take now $\xi \notin L$. The automaton $A$ accepts $\xi$ with probability $p_\xi \leq p_1$. With $\eta_0$ as above, we then have $0 < \eta_0 < \bar{p} - p_1 \leq \bar{p} - p_\xi$ and

$$P\left\{\frac{\mu_m^\xi}{m} > \bar{p}\right\} \leq P\left\{\left|\frac{\mu_m^\xi}{m} - p_\xi\right| \geq \eta_0\right\} \leq \frac{1}{4m\eta_0^2}. \tag{7}$$

By (6) and (7), for every $\varepsilon > 0$, if we take $n > \frac{1}{4\varepsilon\eta_0^2}$, we get a system $A_n$ which recognizes $L$ with interval $(\varepsilon_1, 1-\varepsilon_2)$, where $\varepsilon_1, \varepsilon_2 < \varepsilon$.

Now, simulate $A_n$ by automaton $A' = (Q', \Sigma, q_0', Q_F', \delta')$ with the $n$-th Cartesian power of $Q$ as $Q'$, $q_0' = (q_0, \ldots, q_0)$, $Q_F'$ consisting of elements with more than $n\bar{p}$ entries in $Q_F$, and $\delta'$ determined by the $|Q|^n \times |Q|^n$ matrix $V_\sigma' = \bigotimes_{1 \leq i \leq n} V_\sigma$, the $n$-th tensor power of the transition matrices $V_\sigma$ of $A$, $\sigma \in \Gamma$. The tensor product of doubly stochastic matrices being doubly stochastic, the automaton $A'$ is a PRA-C. □

### 2.2 Languages recognized by PRA-C

**Lemma 1.** *A language PRA-C recognizable with interval $(p_1, p_2)$ is PRA-C recognizable with probability $p$, where $p = \frac{p_2}{p_1+p_2}$ if $p_1 + p_2 \geq 1$, and $p = \frac{1-p_1}{2-p_1-p_2}$ if $p_1 + p_2 < 1$.*

*Proof.* Assume a PRA-C $A$ recognizing a language $L$ with interval $(p_1, p_2)$ has $n-1$ states. Consider first the case $p_1 + p_2 > 1$. Informally, having read the end-marker symbol #, we simulate $A$ with probability $\frac{1}{p_1+p_2}$ and reject input with probability $\frac{p_1+p_2-1}{p_1+p_2}$. Formally, to recognize $L$ with probability $\frac{p_2}{p_1+p_2}$, we modify $A$ by adding a new state $q_r \notin Q_F$, and adjusting the transition function so that $\delta(q_r, \sigma, q_r) = 1$, $\sigma \neq \#$, $\delta(q_0, \#, q_r) = \frac{p_1+p_2-1}{p_1+p_2}$, and, $\delta(q_0, \#, q) = \frac{1}{p_1+p_2}\delta_{old}(q_0, \#, q)$, $q \neq q_r$.

The modified automaton has $n$ states. Since end-marker symbol # is read only once at the beginning of an input word, we can disregard the rest of transition function values associated with #; we put $\delta(q, \#, q') = \frac{1-\delta(q_0,\#,q')}{n-1}$, $q \neq q_0$. The transition function meets the requirements of Definition 4 and the constructed automaton recognizes $L$ with probability $\frac{p_2}{p_1+p_2}$. The case $p_1 + p_2 < 1$ is similar. Informally, having read end-marker symbol #, we simulate $A$ with probability $\frac{1}{2-p_1-p_2}$ and accept input with probability $\frac{1-p_1-p_2}{2-p_1-p_2}$. □

**Lemma 2.** *If two languages are PRA-C recognizable with probability greater than $\frac{2}{3}$ each, then their intersection and union are PRA-C recognizable with probability greater than $\frac{1}{2}$.*

*Proof.* Let $A_i = (Q^i, \Sigma, q_0^i, Q_F^i, \delta^i)$ be PRA-C recognizing language $L_i$ with probability $p_i > \frac{2}{3}$, $i = 1, 2$, and assume without loss of generality that $p_1 \leq p_2$. Informally, having read end-marker symbol $\#$, with probability $\frac{1}{2}$ we simulate the automaton $A_1$ and with the same probability we simulate the automaton $A_2$. Formally, we construct a PRA-C $C = (Q, \Sigma, q_0, Q_F, \delta)$ by first putting $Q = Q^1 \cup Q^2$, $q_0 = q_0^i$, and, $Q_F = Q_F^1 \cup Q_F^2$, and then defining $\delta = \delta^1 \cup \delta^2$ with the exception that $\delta(q_0, \#, q) = \frac{1}{2}\delta^i(q_0, \#, q)$, $q \in Q$, and, $\delta(q', \#, q) = \frac{1}{N}(1 - \delta(q_0, \#, q))$, $N = |Q^1| + |Q^2| - 1$, $q, q' \in Q$, $q \neq q_0$.

Write $\bar{p} = \frac{1}{2}(p_1 + p_2)$. The automaton $C$ recognizes the languages $L_1 \cap L_2$ and $L_1 \cup L_2$ with intervals $(1 - a_1, b_1)$ and $(1 - b_2, a_2)$, respectively, $b_1, b_2 \geq \bar{p}$, $a_1, a_2 \geq \frac{1}{2}p_1$, both intervals being non-empty if $p_1, p_2 > \frac{2}{3}$. The conclusion now follows by Lemma 1. □

**Theorem 2.** *The class of PRA-C recognizable languages is closed under intersection, union and complement.*

*Proof.* Any two PRA-C recognizable languages are by Theorem 1 recognizable with probability $1 - \varepsilon$, and hence by Lemmas 1 and 2, the union and the intersection of these languages are recognizable. If a language $L$ is recognized by a PRA-C $A$, then $\overline{L}$ is recognized by the automaton obtained from $A$ by interchanging its accepting and rejecting states. □

**Theorem 3.** *Languages recognizable by PRA-C with probability* 1 *are recognizable by permutation automata.*

*Proof.* Let $A$ be a PRA-C with state space $Q$ which recognizes a language $L \subset \Sigma^*$ with probability 1. Write $q\omega$ for the set of states accessed by $A$ from state $q$ with positive probability upon reading a word $\omega \in \Sigma^*$. Since $A$ accepts words in $L$ with probability 1, and those not in $L$ with probability 0, we must have either $q\omega \subset Q_F$ or $q\omega \subseteq \overline{Q}_F$ for all $q \in Q, \omega \in \Sigma^*$; write $\widetilde{q\omega} = 1$ in the former case, and $\widetilde{q\omega} = 0$ in the latter. Put $q \sim q'$ iff $\widetilde{q\omega} = \widetilde{q'\omega}$ for all $\omega \in \Sigma^*$; clearly, $\sim$ is an equivalence relation on $Q$. Define now a deterministic automaton $D$ with the set of equivalence classes $[q]$ of $\sim$, $q \in Q$, as state space, transitions $[q]\sigma = [q\sigma]$, $\sigma \in \Sigma$, and accepting states $[q] \subset Q_F$. It should be clear that $D$ simulates $A$, and that its transition matrices are permutation matrices, the transition matrices of $A$ being doubly stochastic. □

**Theorem 4.** *The class of PRA-C recognizable languages is closed under inverse homomorphisms.*

*Proof.* Consider finite alphabets $\Sigma, T$, a homomorphism $h : \Sigma^* \longrightarrow T^*$, a language $L \subseteq T^*$ and a PRA-C $A$, which recognizes $L$. Modify the transition matrices of $A$ by putting $V_\sigma = V_{h(\sigma)}$, $\sigma \in \Sigma$. The modified automaton recognizes $h^{-1}(L)$ with the same interval as $A$. □

**Corollary 1.** *The class of PRA-C recognizable languages is closed under word quotient.*

*Proof.* This follows from closure under inverse homomorphisms and presence of end-markers $\#, \$$. □

We note that the closure property under word quotient remains true also for PRA-C without end-markers; see the Appendix for details.

**Theorem 5.** *For every positive integer $n$, the language $L_n = a_1^* a_2^* \dots a_n^*$ over an alphabet $\{a_1, a_2, \dots, a_n\}$ is PRA-C recognizable.*

*Proof.* We construct a PRA-C $A$ with $n+1$ states; in the construction, we employ the notation $\mathbf{1}_k$ and $1_k$ for the $k \times k$ matrix and the $k \times 1$ vector, respectively, with all entries equal to one. The initial state $q_0$ of $A$ corresponds to the probability distribution vector having 1 as its first entry. The transition function is determined by the matrices

$$V_{a_i} = \frac{1}{i} \cdot \mathbf{1}_i \oplus \frac{1}{n+1-i} \cdot \mathbf{1}_{n+1-i}, \qquad i = 1, \dots, n; \tag{8}$$

explicitly, $V_{a_i}$ has the matrices $i^{-1} \cdot \mathbf{1}_i$ and $(n+1-i)^{-1} \cdot \mathbf{1}_{n+1-i}$ on the diagonal, and remaining entries zero. The accepting states are $q_0 \dots q_{n-1}$, and the only rejecting state is $q_n$.

If $\omega \in L_n$, having read $\omega \in a_1^* \dots a_{k-1}^* a_k^+$, the automaton $A$ is in probability distribution $k^{-1} \cdot 1_k \oplus 0 \cdot 1_{n+1-k}$, and hence $\omega$ is accepted with probability 1.

If $\omega \notin L_n$, consider $k$ such that $\omega = \omega_1 \sigma \omega_2$, $|\omega_1| = k$, $\omega_1 \in L_n$ and $\omega_1 \sigma \notin L_n$; since all one-letter words are in $L_n$, $k$ must be positive. Let $a_t = [\omega]_k$ and $a_s = \sigma$. We have $1 \le s < t \le n$. Having read $\omega_1 \in a_1^* \dots a_{t-1}^* a_t^+$, the automaton is in the distribution $t^{-1} \cdot 1_t \oplus 0 \cdot 1_{n+1-t}$. After that, having read $a_s$, the automaton is in the distribution

$$(\frac{1}{s} \cdot \mathbf{1}_s \oplus \frac{1}{n+1-s} \cdot \mathbf{1}_{n+1-s}) \cdot (\frac{1}{t} \cdot 1_t \oplus 0 \cdot 1_{n+1-t}),$$

which is equal to

$$\frac{1}{t} \cdot 1_s \oplus \frac{t-s}{t(n+1-s)} \cdot 1_{n+1-s}.$$

So the word $\omega_1 a_s$ is accepted with probability $1 - \frac{t-s}{t(n-s+1)}$. Since $\frac{t-s}{t(n-s+1)} < \frac{1}{t}$, reading the symbols succeeding $\omega_1 a_s$ does not increase the accepting probability by Lemma 7. Hence, to find the highest accepting probability $p_1$ for words not in $L_n$ it is enough to maximize $1 - \frac{t-s}{t(n-s+1)}$ over $1 \le s < t \le n$; the maximum is achieved for $(s,t) = (k, k+1)$ if $n = 2k$, and $(s,t) = (k, k+1)$ or $(s,t) = (k+1, k+2)$ if $n = 2k+1$, $k \ge 0$. Hence $p_1 = 1 - \frac{1}{(k+1)^2}$ if $n = 2k$, and $p_1 = 1 - \frac{1}{(k+1)(k+2)}$ if $n = 2k+1$. All in all, $A$ recognizes $L_n$ with interval $\left(1 - \frac{1}{\lfloor (\frac{n}{2})^2 \rfloor + n + 1}, \ 1\right)$, and so, by Theorem 1, $L_n$ is recognizable with probability $1 - \varepsilon$. □

**Corollary 2.** *Quantum finite automata with mixed states (model of Nayak, [N 99]) recognize $L_n = a_1^* a_2^* \dots a_n^*$ with probability $1 - \varepsilon$.*

*Proof.* The matrices in (8) and their tensor powers all have unitary prototypes; see Definition 9. □

### 2.3 Languages not recognized by PRA-C

**Definition 5.** *We say that a regular language is of Type* 0 *if the following is true for the minimal deterministic automaton recognizing this language: there exist states* $q$, $q_1 \neq q_2$ *and words* $x$, $y$ *such that*

1. $qx = q_1$, $qy = q_2$;
2. $q_1x = q_1$, $q_2y = q_2$;
3. $\forall t \in (x,y)^*\ \exists t_1 \in (x,y)^*\ q_1tt_1 = q_1$;
4. $\forall t \in (x,y)^*\ \exists t_2 \in (x,y)^*\ q_2tt_2 = q_2$.

**Definition 6.** *We say that a regular language is of Type* 2 *if the following is true for the minimal deterministic automaton recognizing this language: there exist states* $q$, $q_1 \neq q_2$ *and words* $x$, $y$ *such that*

1. $qx = q_1$, $qy = q_2$;
2. $q_1x = q_1$, $q_1y = q_1$;
3. $q_2x = q_2$, $q_2y = q_2$.

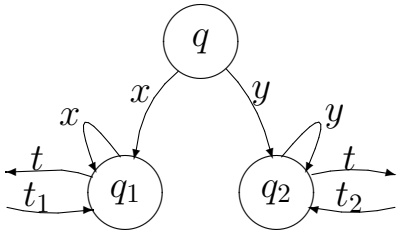


**Fig. 1.** Type 0 construction

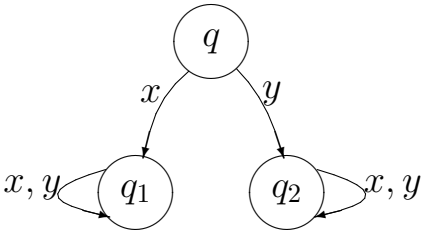


**Fig. 2.** Type 2 construction

**Definition 7.** *We say that a regular language is of Type* 1 *if the following is true for the minimal deterministic automaton recognizing this language: there exist states* $q_1 \neq q_2$ *and words* $x$, $y$ *such that*

1. $q_1x = q_2$, $q_2x = q_2$;
2. $q_2y = q_1$.

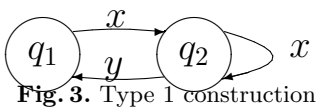


**Fig. 3.** Type 1 construction

Type 1 languages are exactly the languages that violate the partial order condition of [BP 99].

**Lemma 3.** *If $A$ is a deterministic finite automaton with a set of states $Q$ and alphabet $\Sigma$, then $\forall q \in Q\ \forall x \in \Sigma^*\ \exists k > 0\ qx^k = qx^{2k}$.*

*Proof.* We paraphrase a result from the theory of finite semigroups. Consider a state $q$ and a word $x$. Since number of states is finite, $\exists m \geq 0\ \exists s \geq 1\ \forall n\ qx^m = qx^m x^{sn}$. Take $n_0$, such that $sn_0 > m$. Note that $\forall t \geq 0\ qx^{m+t} = qx^{m+t}x^{sn_0}$. We take $t = sn_0 - m$, so $qx^{sn_0} = qx^{sn_0}x^{sn_0}$. Take $k = sn_0$. □

**Lemma 4.** *A regular language is of Type 0 iff it is of Type 1 or Type 2.*

*Proof.* It is obvious that a language of Type 2 is also of Type 0. Take now a language $L$ of Type 1 with states $q_1'', q_2''$ and words $x'', y''$. To obtain a construction of Type 0, take $q = q_1 = q_1''$, $q_2 = q_2''$, $x = x''y''$, $y = x''$. That forms transitions $qx = q_1$, $qy = q_2$, $q_1x = q_1$, $q_1y = q_2$, $q_2x = q_1$, $q_2y = q_2$. Hence $L$ is of Type 0. Finally, consider a language $L$ of Type 0. By Lemma 3,

$\exists t \exists b\ q_1y^b = q_t$ and $q_ty^b = q_t$;
$\exists u \exists c\ q_2x^c = q_u$ and $q_ux^c = q_u$.

If $q_1 \neq q_t$ then by the 3rd rule of Type 0, $\exists z\ q_tz = q_1$, and so $L$ is of Type 1. If $q_2 \neq q_u$ then by the 4th rule of Type 0, $\exists z\ q_uz = q_2$, and $L$ is of Type 1. If $q_1 = q_t$ and $q_2 = q_u$, we have $qx^c = q_1$, $qy^b = q_2$, $q_1x^c = q_1y^b = q_1$, $q_2x^c = q_2y^b = q_2$; we get the construction of Type 2 by taking $x' = x^c$, $y' = y^b$. □

**Lemma 5.** *Regular languages of Type 2 are not PRA-C recognizable.*

*Proof.* Assume $A$ is a PRA-C recognizing a language $L \subset \Sigma^*$ of Type 2. Being of Type 2, the language $L$ is recognized by a deterministic automaton $D$ with particular three states $q$, $q_1$, $q_2$ such that $q_1 \neq q_2$, $qx = q_1$, $qy = q_2$, $q_1x = q_1$, $q_1y = q_1$, $q_2x = q_2$, $q_2y = q_2$, where $x, y \in \Sigma^*$. Furthermore, there exists $\omega \in \Sigma^*$ such that $q_0\omega = q$, where $q_0$ is the initial state of $D$, and there exists $z \in \Sigma^*$, such that $q_1z$ is an accepting state and $q_2z$ is a rejecting state of $D$.

Denote the transition matrices of $A$ corresponding to the words $x, y, \omega, z$ by $X, Y, W, Z$, respectively. By Theorem 12, there exist an positive integer $k$ such that every state of $Q$ is accessible from itself by $X^k$ and by $Y^k$. Consequently, by Corollary 6 with $S = (X^k, Y^k)^*$, the accessibility relation "$q \to q'$ by *some* $T \in S$" is an equivalence relation on $Q$, coinciding with the relation "$q \to q'$ by $X^kY^k$". Put $C = Y^kX^k$ and renumber the states if necessary; the matrix $C$ is then by Theorem 12 block diagonal, each block corresponding to an aperiodic irreducible doubly stochastic Markov chain with states in the equivalence class of its accessibility relation. It follows by Corollary 5 that the limit of $C^m$ as $m \to \infty$ exists and is a block diagonal matrix $J$ with blocks of form $\frac{1}{p} \cdot \mathbf{1}_p$. The accessibility by $Y^k$ being a refinement of the relation of accessibility by $C$, the matrix $Y^k$ is also block diagonal with blocks refining those of $C$ and $J$. So $JY^K = J$, and the limits of $Z(Y^KX^K)^mW$ and $Z(Y^KX^K)^mY^KW$ as $m \to \infty$ both equal $ZJW$. However, by construction of Type 2, $\omega(x^ky^k)^mz \in L$ and $\omega y^k(x^ky^k)^mz \notin L$, for all $k$ and $m$. This is a contradiction. □

**Lemma 6.** *Regular languages of Type* 1 *are not PRA-C recognizable.*

*Proof.* The proof parallels that of Lemma 5. Consider a PRA-C $A$ which recognizes a language $L$ of Type 1. For the words $x, y$ there is a constant $k$, such that the probabilities of acceptance by $A$ of the words $\xi_1 = \omega(x^k(xy)^k)^m z$ and $\xi_2 = \omega(x^k(xy)^k)^m x^k z$ are for large $m$ arbitrarily close. But we can choose $z$ so that $\xi_1 \in L$ and $\xi_2 \notin L$. □

**Theorem 6.** *Regular languages of Type* 0 *are not PRA-C recognizable.*

*Proof.* By Lemmas 4, 5, 6. □

We note by Lemma 4 that the construction of Type 0 generalizes a construction proposed by [BP 99]. It is also easily noticed, that Type 0 construction generalizes a construction proposed by [AKV 00]. The constructions of [BP 99] and [AKV 00] characterize languages not recognized by measure-many quantum finite automata of [KW 97].

**Corollary 3.** *The languages (a,b)*a and a(a,b)* are not PRA-C recognizable.*

*Proof.* Both languages are of Type 0. □

**Corollary 4.** *The class of languages recognizable by PRA-C is not closed under homomorphisms.*

*Proof.* Consider a homomorphism $\{a, b, c\}^* \to \{a, b\}^*$ which leaves $a$ and $b$ unchanged and sends $c$ to $a$. Reasoning as in the proof of Theorem 5, the language $(a, b)^* cc^*$ is recognizable by a PRA-C. (Take $n = 2$, $V_a = V_{a_1}$, $V_b = V_{a_1}$, $V_c = V_{a_2}$, and $Q_F = \{q_1\}$.) However, by Corollary 3, the language $(a, b)^* aa^* = (a, b)^* a$ is not recognizable. □

## 3 1-way probabilistic reversible DH-automata

These automata are defined as in Definition 4, except that now the languages are recognized in the decide-and-halt sense of Definition 2. The class of languages recognized by PRA-C is a proper subclass of the languages recognized by PRA-DH: the language $a(a, b)^*$, for example, is PRA-DH recognizable. However,

**Theorem 7.** *The language (a,b)*a is not PRA-DH recognizable.*

*Proof.* A PRA-DH automaton reading a sequence of $a$:s and $b$:s can halt only with some probability $p$ strictly less then one, so accepting and rejecting probabilities may differ only by $1 - p$, because any word belonging to the language is not dependent on any prefix. Therefore for each $\varepsilon > 0$ we can find that after reading of a prefix of certain length, the total probability to halt while continue reading the word is less then $\varepsilon$. In this case we reason as in the proof of Lemma 5: for words $x, y$ take a constant $k$ such that the probabilities of acceptance of the words $\omega(x^k(xy)^k)^m z$ and $\omega(x^k(xy)^k)^m x^k z$ are arbitrarily close for large $m$. □

## 4 Alternative approach to finite reversible automata and 1.5-way probabilistic reversible automata

Consider the transpose $A^t$ of a probabilistic automaton $A$ obtained by transposing $A$'s transfer function: $\delta^t(q_1, \sigma, q_2) = \delta(q_2, \sigma, q_1)$. If $A^t$ is a valid probabilistic automaton, we may see $A$ and $A^t$ as probabilistic reversible automata. Generally, we propose to call an automaton $A$ of some type **weakly reversible** if its transpose $A^t$ is an automaton of the same type. Note that the transpose of a deterministic automaton (with transfer function taking values in the set $\{0, 1\}$) is still a deterministic automaton, not nondeterministic.

For 1-way probabilistic automata, the notions of weak reversibility and reversibility (the requirement that its transition matrix be doubly stochastic as articulated in Definition 4), coincide. However, for 1.5-way probabilistic automata about to be defined, the two notions differ.

**Definition 8.** [1.5-way PRA-C] *A 1.5-way finite* ***probabilistic automaton*** *(PA) is specified by a finite set of states $Q$, a finite input alphabet $\Sigma$, and a transition function*

$$\delta : Q \times \Gamma \times Q \times D \longrightarrow \mathbb{R}_{[0,1]},$$

*with $\Gamma$ as in Definition 4 of 1-way PA, and $D = \{0, 1\}$ indicating whether automaton stays on the same position or moves one letter ahead on the input tape. For all $q \in Q$ and $\sigma \in \Gamma$, the transition function is required to satisfy the condition:*

$$\sum_{q' \in Q, d \in D} \delta(q, \sigma, q', d) = 1. \tag{9}$$

*A 1.5-way PA is a probabilistic* ***weakly reversible*** *automaton (PwRA) if for all $q \in Q$ and $\sigma \in \Gamma$,*

$$\sum_{q' \in Q, d \in D} \delta(q', \sigma, q, d) = 1. \tag{10}$$

*The 1,5-way PA is* ***reversible*** *(PRA) if for all $q \in Q$ and $\sigma_1, \sigma_2 \in \Gamma$,*

$$\sum_{q' \in Q} \delta(q', \sigma_1, q, 0) + \sum_{q' \in Q, \sigma \in \Gamma} \delta(q', \sigma_2, q, 1) = 1. \tag{11}$$

*A 1.5-way PRA-C (PwRA-C) is a 1.5-way PRA (PwRA-C) with classical word acceptance (Definition 1): there is an initial state $q_0 \in Q$, in which the processing of all words begins, and a set of accepting states $Q_F \subseteq Q$, which determine whether the word is accepted upon completed processing of the word's last letter.*

**Theorem 8.** *The language (a,b)*a is recognizable by a 1.5-way weakly reversible PRA-C.*

*Proof.* The $Q = \{q_0, q_1\}$, $Q_F = \{q_1\}$, $\delta$ is defined as follows
$\delta(q_0, a, q_0, 0) = \frac{1}{2}$ $\delta(q_0, a, q_1, 1) = \frac{1}{2}$ $\delta(q_1, a, q_0, 0) = \frac{1}{2}$ $\delta(q_1, a, q_1, 1) = \frac{1}{2}$
$\delta(q_0, b, q_0, 1) = \frac{1}{2}$ $\delta(q_0, b, q_1, 0) = \frac{1}{2}$ $\delta(q_1, b, q_0, 1) = \frac{1}{2}$ $\delta(q_1, b, q_1, 0) = \frac{1}{2}$
$\delta(q_0, \$, q_0, 1) = 1$ $\delta(q_1, \$, q_1, 1) = 1$

It is easy to check that such automaton moves ahead according to the transition of the following deterministic automaton
$\delta(q_0, a, q_1, 1) = 1$ $\delta(q_1, a, q_1, 1) = 1$
$\delta(q_0, b, q_0, 1) = 1$ $\delta(q_1, b, q_0, 1) = 1$
$\delta(q_0, \$, q_0, 1) = 1$ $\delta(q_1, \$, q_1, 1) = 1$

So the probability of wrong answer is 0. The probability to be at the $m$-th position of the input tape after $n$ steps of calculation for $m \leq n$ is $C_n^m$. Therefore it is necessary no more than $O(n * \log(p))$ steps to reach the end of the word of length n (and so obtain correct answer) with probability $1 - \frac{1}{p}$. □

## 5 A classification of reversible automata

We propose the following classification of finite 1-way reversible automata:

| | C-automata | DH-automata |
|---|---|---|
| Deterministic Automata | Permutation Automata [HS 66,T 68] (DRA-C) | Reversible Finite Automata [AF 98] (DRA-DH) |
| Quantum Automata with Pure States | Measure-Once Quantum Finite Automata [MC 97] (QRA-P-C) | Measure-Many Quantum Finite Automata [KW 97] (QRA-P-DH) |
| Probabilistic Automata | Probabilistic Reversible C-automata (PRA-C) | Probabilistic Reversible DH-automata (PRA-DH) |
| Quantum Finite Automata with Mixed States | not considered yet (QRA-M-C) | Enhanced Quantum Finite Automata [N 99] (QRA-M-DH) |

Language class problems have been solved for DRA-C, DRA-DH, QRA-P-C, for the remaining types they are still open. Every type of DH-automata may simulate the corresponding type of C-automata. Generally, language classes recognized by C-automata are closed under boolean operations (though this is open for QRA-M-C), while DH-automata are not (though this is open for QRA-M-DH and possibly for PRA-DH).

**Definition 9.** *We say that a unitary matrix $U$ is a* prototype *for a doubly stochastic matrix $S$, if $|U_{ij}|^2 = S_{ij}$ $(\forall i, j)$.*

Note that not every doubly stochastic matrix has a unitary prototype; take for example, the three-by-three matrix with zeros along the dexter diagonal and all remaining entries equal to $\frac{1}{2}$.

In the Introduction, we touched upon the relation between PRA-C and QRA-M-DH (and hence, QRA-M-C). Due to the example above, we do not know

whether every PRA-C can be simulated by QRA-M-C, or whether every PRA-DH can be simulated by QRA-M-DH. The following results are however straightforward.

**Theorem 9.** *If all matrices of a PRA-C have unitary prototypes, then the PRA-C may be simulated by a QRA-M-C and by a QRA-M-DH.*

**Theorem 10.** *If all matrices of a PRA-DH have unitary prototypes, then the PRA-DH may be simulated by a QRA-M-DH.*

## 6 Appendix A: Doubly stochastic Markov chains

We briefly recall some background material on finite Markov chains; see, for example, [KS 76]. A Markov chain with $n$ states $q_1, \ldots, q_n$ is determined by an $n \times n$ (column-) stochastic matrix $A$, ie a matrix with non-negative entries $A_{ij}$, in which the elements of every column sum up to one. If $A_{ij} = p > 0$ then a state $q_i$ is *accessible* from a state $q_j$ *in one step* with a positive probability $p$. Of course, the matrix of Markov chain depends on the numbering of the states; if the states are renumbered, its rows and columns must also be renumbered.

A state $q_j$ is *accessible* from $q_i$ (denoted $q_i \to q_j$) if there is a positive probability to get from $q_i$ to $q_j$ (possibly in several steps). States $q_i$ and $q_j$ *communicate* (denoted $q_i \leftrightarrow q_j$) if $q_i \to q_j$ and $q_j \to q_i$. A state $q$ is *ergodic* if $\forall i\ q \to q_i \Rightarrow q_i \to q$; otherwise the state is called *transient*. A Markov chain without transient states is called *irreducible* if for all $q_i, q_j\ q_i \leftrightarrow q_j$; otherwise the chain is *reducible*. The *period* of an ergodic state $q_i \in Q$ of a Markov chain with a matrix $A$ is defined as $d(q_i) = \gcd\{n > 0 \mid (A^n)_{ii} > 0\}$. An ergodic state $q_i$ is called *aperiodic* if $d(q_i) = 1$; otherwise it is *periodic*. A Markov chain without transient states is called *aperiodic* if all its states are aperiodic; otherwise the chain is called *periodic*. A probability distribution X of a Markov chain with a matrix $A$ is called *stationary*, if $AX = X$. A Markov chain is called *doubly stochastic*, if its transition matrix is a doubly stochastic matrix.

**Theorem 11.** *An irreducible and aperiodic Markov chain with matrix $A$ has a unique stationary probability distribution $Z$, and $A^n \to (Z, \ldots, Z)$ as $n \to \infty$; consequently, $A^n X \to Z$ as $n \to \infty$ for all probability distribution vectors $X$.*

**Corollary 5.** *If a doubly stochastic Markov chain with an $m \times m$ matrix $A$ is irreducible and aperiodic, then $A^n \to \frac{1}{m} \cdot \mathbf{1}_m$ as $n \to \infty$, and, consequently, $A^n X \to \frac{1}{m} \cdot 1_m$ as $n \to \infty$, for all probability distribution vectors $X$; here $\mathbf{1}_m$ and $1_m$ stand for the $m \times m$ matrix and the $m \times 1$ vector, respectively, with all entries equal to one.*

A square matrix $A$ is *irreducible* if there is no permutation matrix $P$ such that $P^{-1}AP$ consists of two proper square sub-matrices along the diagonal, all remaining entries above the diagonal being zero. A square matrix $A$ is *completely reducible* if there is a permutation matrix $P$ such that $P^{-1}AP$ is block diagonal with irreducible blocks. Note that a completely reducible matrix may be irreducible.

**Theorem 12.** *The transition matrix of a finite doubly stochastic Markov chain is completely reducible. Equivalently, the accessibility relation $q \to q'$, $q, q' \in Q$, is an equivalence relation on the state space $Q$.*

*Proof.* This is a well known fact; an elementary argument for the second statement is included for completeness.

The transitivity of $\to$ holds trivially for any Markov chain. For reflexivity, assume there is a state $q_0$ such that the set $Q_0$ of all states accessible from $q_0$ does not contain $q_0$. Say $Q_{q_0}$ has $k$ elements; clearly $k > 0$. Let $A'$ be the submatrix of $A$ indexed by the states in $Q_{q_0}$. A column $j$ of $A'$ must include all non-zero elements of the corresponding column of $A$ since, by the transitivity of $\to$, the states accessible from $q_j \in Q_0$ are accessible from $q_0$. Hence $A'$ is column-stochastic, and summing its entries 'columns first' yields $k$. On the other hand, a row of $A'$ indexed by a state accessible in one step from $q_0$ does not include all the nonzero elements of the corresponding row of $A$. Since A is row-stochastic, summing the entries of $A'$ 'rows first' yields a number strictly less than $k$. This is a contradiction, hence $\to$ is reflexive. For the symmetry of $\to$, write $q \to q'$ as a sequence of one-step transitions, show the symmetry of each of these by rephrasing the above argument, and use transitivity. □

Hence, a doubly stochastic Markov chain has no transient states; it is either periodic or aperiodic, either reducible or irreducible.

**Corollary 6.** *Let $S$ be a semi-group of doubly stochastic matrices acting on a state space $Q$. Then the accessibility relation "$q \to q'$ by some $A \in S$" is an equivalence relation on $Q$. Furthermore, if $S$ is generated by a finite number of matrices $A_1, \ldots, A_N$ under each of which every state in $Q$ is accessible in one step from itself, then the product matrix $A = A_1 \cdot \ldots \cdot A_N$ induces the same accessibility relation on $Q$ as the semigroup $S$.*

*Proof.* Only the second statement needs an argument, and only in one direction. Let $q'$ be accessible from $q$ by an element $A_1^{k_1} \cdot \ldots \cdot A_N^{k_N}$ of $S$ in $m$ steps. For every step, there is a sequence of states $q_1, q_2, \ldots, q_{N+1}$, such that $q_i \to q_{i+1}$ by $A_i^{k_i}$, and hence also by $A^{k_i}$ in one step, since every state is accessible from itself in one step by every $A_i$, $i = 1, \ldots N$. If follows that $q \to q'$ by $A$ in $m \cdot \sum_i k_i$ steps. □

For completeness, we end this Section with an elementary lemma used in the proof of Theorem 5.

**Lemma 7.** *Let $A$ be a finite row-stochastic matrix. Then, for all real vectors $X$, all entries of the product $AX$ are contained in the interval* $[\min(X), \max(X)]$.

*Proof.* For a matrix $A$ with non-negative entries, the vector 1 with all entries equal to one is an eigenvector of $A$ with eigenvalue one if (and only if) $A$ is row-stochastic. The conclusion now follows from the defining inequality $\min(X) \cdot 1 \leq X \leq \max(X) \cdot 1$.

## 7 Appendix B: End-marker theorems for PRA-C

We denote a PRA-C with both end-markers as #,$-PRA-C. We denote a PRA-C with left end-marker only as #-PRA-C.

**Theorem 13.** *Let $A$ be a #,$-PRA-C, which recognizes a language $L$. There exists a #-PRA-C which recognizes the same language.*

*Proof.* Suppose $A = (Q, \Sigma, q_0, Q_F, \delta)$, where $|Q| = n$. $A$ recognizes $L$ with interval $(p_1, p_2)$. We construct the following automaton $A' = (Q', \Sigma, q_{0,0}, Q'_F, \delta')$ with $mn$ states. Informally, $A'$ equiprobably simulates $m$ copies of the automaton $A$.

$Q' = \{q_{0,0}, \ldots, q_{0,m-1}, q_{1,0}, \ldots, q_{1,m-1}, \ldots, q_{n-1,0}, \ldots, q_{n-1,m-1}\}$.

If $\sigma \neq \#$, $\delta'(q_{i,k}, \sigma, q_{j,l}) = \begin{cases} \delta(q_i, \sigma, q_j), \text{ if } k = l \\ 0, \text{ if } k \neq l. \end{cases}$

Otherwise, $\delta'(q_{0,0}, \#, q_{j,l}) = \frac{1}{m}\delta(q_0, \#, q_j)$, and if $q_{i,k} \neq q_{0,0}$, $\delta'(q_{i,k}, \#, q) = \frac{1-\delta'(q_{0,0}, \#, q)}{mn-1}$. Function $\delta'$ satisfies the requirements (1) and (2) of Definition 4.

We define $Q'_F$ as follows. A state $q_{i,k} \in Q'_F$ if and only if $0 \leq k < mp(q_i)$, where $p(q_i) \stackrel{\text{def}}{=} \sum\limits_{q \in Q_F} \delta(q_i, \$, q)$.

Suppose $\#\omega\$$ is an input word. Having read $\#\omega$, $A$ is in superposition $\sum\limits_{i=0}^{n-1} a_i^\omega q_i$. After $A$ has read \$, $\#\omega\$$ is accepted with probability $p_\omega = \sum\limits_{i=0}^{n-1} a_i^\omega p(q_i)$.

On the other hand, having read $\#\omega$, $A'$ is in superposition $\frac{1}{m} \sum\limits_{j=0}^{m-1} \sum\limits_{i=0}^{n-1} a_i^\omega q_{i,j}$. So the input word $\#\omega$ is accepted with probability $p'_\omega = \frac{1}{m} \sum\limits_{i=0}^{n-1} a_i^\omega \lceil mp(q_i) \rceil$.

Consider $\omega \in L$. Then $p'_\omega = \frac{1}{m} \sum\limits_{i=0}^{n-1} a_i^\omega \lceil mp(q_i) \rceil \geq \sum\limits_{i=0}^{n-1} a_i^\omega p(q_i) = p_\omega \geq p_2$.

Consider $\xi \notin L$. Then $p'_\xi = \frac{1}{m} \sum\limits_{i=0}^{n-1} a_i^\xi \lceil mp(q_i) \rceil < \sum\limits_{i=0}^{n-1} a_i^\xi p(q_i) + \frac{1}{m} \sum\limits_{i=0}^{n-1} a_i^\xi = p_\xi + \frac{1}{m} \leq p_1 + \frac{1}{m}$.

Therefore $A'$ recognizes $L$ with bounded error, provided $m > \frac{1}{p_2 - p_1}$. □

Now we are going to prove that PRA-C without end-markers recognize the same languages as #-PRA-C automata.

If $A$ is a #-PRA-C, then, having read the left end-marker #, the automaton simulates some other automata $A_0, A_1, \ldots, A_{m-1}$ with positive probabilities $p_0, \ldots, p_{m-1}$, respectively. $A_0, A_1, \ldots, A_{m-1}$ are automata without end-markers. By $p_{i,\omega}$, $0 \leq i < m$, we denote the probability that the automaton $A_i$ accepts the word $\omega$.

We prove the following lemma first.

**Lemma 8.** *Suppose $A'$ is a #-PRA-C which recognizes a language $L$ with interval $(a_1, a_2)$. Then for every $\varepsilon$, $0 < \varepsilon < 1$, exists a #-PRA-C $A$ which recognizes $L$ with interval $(a_1, a_2)$, such that*

*a) if $\omega \in L$, $p_{0,\omega} + p_{1,\omega} + \ldots + p_{n-1,\omega} > \frac{a_2 n}{1+\varepsilon}$*

*b) if* $\omega \notin L$, $p_{0,\omega} + p_{1,\omega} + \ldots + p_{n-1,\omega} < \frac{a_1 n}{1-\varepsilon}$.

*Here* $n$ *is the number of automata without end-markers, being simulated by* $A$*, and* $p_{i,\omega}$ *is the probability that* $i$*-th simulated automaton* $A_i$ *accepts* $\omega$*.*

*Proof.* Suppose a #-PRA-C $A'$ recognizes a language $L$ with interval $(a_1, a_2)$. Having read the symbol #, $A'$ simulates automata $A'_0, \ldots, A'_{m-1}$ with probabilities $p'_0, \ldots, p'_{m-1}$, respectively. We choose $\varepsilon$, $0 < \varepsilon < 1$.

By Dirichlet's principle ([HW 79], p. 170), $\forall \varphi > 0$ exists $n \in \mathbb{N}^+$ such that $\forall i$ $p'_i n$ differs from some positive integer by less than $\varphi$.

Let $0 < \varphi < \min\left(\frac{1}{m}, \varepsilon\right)$. Let $g_i$ be the nearest integer of $p'_i n$. So $|p'_i n - g_i| < \varphi$ and $\left|\frac{p'_i}{g_i} - \frac{1}{n}\right| < \frac{\varphi}{n g_i} \leq \frac{\varphi}{n}$. Since $|p'_i n - g_i| < \varphi$, we have $\left|n - \sum\limits_{i=0}^{m-1} g_i\right| < \varphi m < 1$. Therefore, since $g_i \in \mathbb{N}^+$, $\sum\limits_{i=0}^{m-1} g_i = n$.

Now we construct the #-PRA-C $A$, which satisfies the properties expressed in Lemma 8. For every $i$, we make $g_i$ copies of $A'_i$. Having read #, for every $i$ $A$ simulates each copy of $A'_i$ with probability $\frac{p'_i}{g_i}$. The construction of $V_{\#}$ is equivalent to that used in the proof of Lemma 2. Therefore $A$ is characterized by doubly stochastic matrices. $A$ recognizes $L$ with the same interval as $A'$, i.e., $(a_1, a_2)$.

Using new notations, $A$ simulates $n$ automata $A_0, A_1, \ldots, A_{n-1}$ with probabilities $p_0, p_1, \ldots, p_{n-1}$, respectively. Note that $\forall i$ $\left|p_i - \frac{1}{n}\right| < \frac{\varphi}{n}$. Let $p_{i,\omega}$ be the probability that $A_i$ accepts the word $\omega$.

Consider $\omega \in L$. We have $p_0 p_{0,\omega} + p_1 p_{1,\omega} + \ldots + p_{n-1} p_{n-1,\omega} \geq a_2$. Since $p_i < \frac{1+\varphi}{n}$, $\frac{1+\varphi}{n}(p_{0,\omega} + p_{1,\omega} + \ldots + p_{n-1,\omega}) > a_2$. Hence

$$p_{0,\omega} + p_{1,\omega} + \ldots + p_{n-1,\omega} > \frac{a_2 n}{1+\varphi} > \frac{a_2 n}{1+\varepsilon}.$$

Consider $\xi \notin L$. We have $p_0 p_{0,\xi} + p_1 p_{1,\xi} + \ldots + p_{n-1} p_{n-1,\xi} \leq a_1$. Since $p_i > \frac{1-\varphi}{n}$, $\frac{1-\varphi}{n}(p_{0,\xi} + p_{1,\xi} + \ldots + p_{n-1,\xi}) < a_1$. Hence

$$p_{0,\xi} + p_{1,\xi} + \ldots + p_{n-1,\xi} < \frac{a_1 n}{1-\varphi} < \frac{a_1 n}{1-\varepsilon}.$$

□

**Theorem 14.** *Let* $A$ *be a #-PRA-C, which recognizes a language* $L$*. There exists a PRA-C without end-markers, which recognizes the same language.*

*Proof.* Consider a #-PRA-C which recognizes a language $L$ with interval $(a_1, a_2)$. Using Lemma 8, we choose $\varepsilon$, $0 < \varepsilon < \frac{a_2 - a_1}{a_2 + a_1}$, and construct an automaton $A'$ which recognizes L with interval $(a_1, a_2)$, with the following properties.

Having read #, $A'$ simulates $A'_0, \ldots, A'_{m-1}$ with probabilities $p'_0, \ldots, p'_{m-1}$, respectively. $A'_0, \ldots, A'_{m-1}$ are automata without end-markers. $A'_i$ accepts $\omega$ with probability $p'_{i,\omega}$. If $\omega \in L$, $p'_{0,\omega} + p'_{1,\omega} + \ldots + p'_{m-1,\omega} > \frac{a_2 m}{1+\varepsilon}$. Otherwise, if $\omega \notin L$, $p'_{0,\omega} + p'_{1,\omega} + \ldots + p'_{m-1,\omega} < \frac{a_1 m}{1-\varepsilon}$.

That also implies that for every $n = km$, $k \in \mathbb{N}^+$, we are able to construct a #-PRA-C $A$ which recognizes $L$ with interval $(a_1, a_2)$, such that

a) if $\omega \in L$, $p_{0,\omega} + p_{1,\omega} + \ldots + p_{n-1,\omega} > \frac{a_2 n}{1+\varepsilon}$;
b) if $\omega \notin L$, $p_{0,\omega} + p_{1,\omega} + \ldots + p_{n-1,\omega} < \frac{a_1 n}{1-\varepsilon}$.

$A$ simulates $A_0, \ldots, A_{n-1}$. Let us consider the system $F_n = (A_0, \ldots, A_{n-1})$. Let $\delta = \frac{1}{2}(a_1 + a_2)$. Since $\varepsilon < \frac{a_2 - a_1}{a_2 + a_1}$, $\frac{a_2}{1+\varepsilon} > \delta$ and $\frac{a_1}{1-\varepsilon} < \delta$. As in the proof of Theorem 1, we define that the system accepts a word, if more than $n\delta$ automata in the system accept the word.

Let us take $\eta_0$, such that $0 < \eta_0 < \frac{a_2}{1+\varepsilon} - \delta < \delta - \frac{a_1}{1-\varepsilon}$.

Consider $\omega \in L$. We have that $\sum\limits_{i=0}^{n-1} p_{i,\omega} > \frac{a_2 n}{1+\varepsilon} > n\delta$. As a result of reading $\omega$, $\mu_n^\omega$ automata in the system accept the word, and the rest reject it. The system has accepted the word, if $\frac{\mu_n^\omega}{n} > \delta$. Since $0 < \eta_0 < \frac{a_2}{1+\varepsilon} - \delta < \frac{1}{n} \sum\limits_{i=0}^{n-1} p_{i,\omega} - \delta$, we have

$$P\left\{\frac{\mu_n^\omega}{n} > \delta\right\} \geq P\left\{\left|\frac{\mu_n^\omega}{n} - \frac{1}{n}\sum_{i=0}^{n-1} p_{i,\omega}\right| < \eta_0\right\}. \tag{12}$$

If we look on $\frac{\mu_n^\omega}{n}$ as a random variable $X$, $E(X) = \frac{1}{n} \sum\limits_{i=0}^{n-1} p_{i,\omega}$ and variance $V(X) = \frac{1}{n^2} \sum\limits_{i=0}^{n-1} p_{i,\omega}(1 - p_{i,\omega})$, therefore Chebyshev's inequality yields the following:

$$P\left\{\left|\frac{\mu_n^\omega}{n} - \frac{1}{n}\sum_{i=0}^{n-1} p_{i,\omega}\right| \geq \eta_0\right\} \leq \frac{1}{n^2\eta_0^2} \sum_{i=0}^{n-1} p_{i,\omega}(1 - p_{i,\omega}) \leq \frac{1}{4n\eta_0^2}.$$

That is equivalent to $P\left\{\left|\frac{\mu_n^\omega}{n} - \frac{1}{n} \sum\limits_{i=0}^{n-1} p_{i,\omega}\right| < \eta_0\right\} \geq 1 - \frac{1}{4n\eta_0^2}$. So, taking into account (12),

$$P\left\{\frac{\mu_n^\omega}{n} > \delta\right\} \geq 1 - \frac{1}{4n\eta_0^2}. \tag{13}$$

On the other hand, consider $\xi \notin L$. So $\sum\limits_{i=0}^{n-1} p_{i,\xi} < \frac{a_1 n}{1-\varepsilon} < n\delta$. Again, since $0 < \eta_0 < \delta - \frac{a_1}{1-\varepsilon} < \delta - \frac{1}{n} \sum\limits_{i=0}^{n-1} p_{i,\xi}$,

$$P\left\{\frac{\mu_n^\xi}{n} > \delta\right\} \leq P\left\{\left|\frac{\mu_n^\xi}{n} - \frac{1}{n}\sum_{i=0}^{n-1} p_{i,\xi}\right| \geq \eta_0\right\} \leq \frac{1}{4n\eta_0^2}. \tag{14}$$

The constant $\eta_0$ does not depend on $n$ and $n$ may be chosen sufficiently large. Therefore, by (13) and (14), the system $F_n$ recognizes $L$ with bounded error, if $n > \frac{1}{2\eta_0^2}$.

Following a way identical to that used in the proof of Theorem 1, it is possible to construct a single PRA-C without end-markers, which simulates the system $F_n$ and therefore recognizes the language $L$. □

## References

[ABFK 99] A. Ambainis, R. Bonner, R. Freivalds and A. Ķikusts. Probabilities to Accept Languages by Quantum Finite Automata. *COCOON 1999, Lecture Notes in Computer Science*, 1999, Vol. 1627, pp. 174-183.
http://arxiv.org/abs/quant-ph/9904066

[AF 98] A. Ambainis and R. Freivalds. 1-Way Quantum Finite Automata: Strengths, Weaknesses and Generalizations. *Proc. 39th FOCS*, 1998, pp. 332-341.
http://arxiv.org/abs/quant-ph/9802062

[AKV 00] A. Ambainis, A. Ķikusts and M. Valdats. On the Class of Languages Recognizable by 1-Way Quantum Finite Automata. *STACS 2001, Lecture Notes in Computer Science*, 2001, Vol. 2010, pp. 75-86.
http://arxiv.org/abs/quant-ph/0009004

[BP 99] A. Brodsky and N. Pippenger. Characterizations of 1-Way Quantum Finite Automata.
http://arxiv.org/abs/quant-ph/9903014

[GS 97] C. M. Grinstead and J. L. Snell. Introduction to Probability. *American Mathematical Society*, 1997.
http://www.dartmouth.edu/~chance/teaching_aids/articles.html

[HW 79] G. H. Hardy and E. M. Wright. An Introduction to the Theory of Numbers. Fifth Edition. *Oxford University Press*, 1979.

[HS 66] J. Hartmanis and R. E. Stearns. Algebraic Structure Theory of Sequential Machines. *Prentice Hall*, 1966.

[KS 76] J. G. Kemeny and J. L. Snell. Finite Markov Chains. *Springer Verlag*, 1976.

[KW 97] A. Kondacs and J. Watrous. On The Power of Quantum Finite State Automata. *Proc. 38th FOCS*, 1997, pp. 66-75.

[MC 97] C. Moore and J. P. Crutchfield. Quantum Automata and Quantum Grammars. *Theoretical Computer Science*, 2000, Vol. 237(1-2), pp. 275-306.
http://arxiv.org/abs/quant-ph/9707031

[N 99] A. Nayak. Optimal Lower Bounds for Quantum Automata and Random Access Codes. *Proc. 40th FOCS*, 1999, pp. 369-377.
http://arxiv.org/abs/quant-ph/9904093

[R 63] M. O. Rabin. Probabilistic Automata. *Information and Control*, 1963, Vol. 6(3), pp. 230-245.

[T 68] G. Thierrin. Permutation Automata. *Mathematical Systems Theory*, 1968, Vol. 2(1), pp. 83-90.

# On the Size of Finite Probabilistic and Quantum Automata

Richard Bonner[1], Rūsiņš Freivalds[2] and Zigmārs Rasščevskis[2]

[1] Department of Mathematics and Physics,
Mälardalen University, Sweden*
richard.bonner@mdh.se
[2] Institute of Mathematics and Computer Science,
University of Latvia, Latvia**
Rusins.Freivalds@mii.lu.lv

**Abstract.** Rabin [Rab63] proved that any language accepted by a probabilistic automaton with an isolated cutpoint is also accepted by some deterministic automaton. He showed, furthermore, that a probabilistic automaton with $n$ states, $r$ accepting states, and radius of isolation $\delta$, has an equivalent deterministic automaton with no more than $(1+\frac{r}{\delta})^n$ states. However, it remained an open problem to find a concrete language for which the deterministic automaton would really need such a large number of states. Freivalds [Fre82] succeeded in constructing a probabilistic automaton with $n$ states, for which the smallest equivalent deterministic automaton requires $\Omega(2^{\sqrt{n}})$ states, and, using his automaton as subroutine, Ambainis [Amb96] constructed a probabilistic automaton with $n$ states for which the smallest equivalent deterministic automaton has $\Omega(2^{\frac{n \log\log n}{\log n}})$ states. Rasščevskis [Ras00] constructed a probabilistic automaton with $n$ states, for which any equivalent deterministic automaton has $\Omega(2^n)$ states. We have found that Rasščevskis' construction can be adapted to prove size advantages of quantum finite automata over the deterministic ones. This new construction turns out to be slightly better than a previously proved result by Ambainis and Freivalds [AF98].

## 1 Introduction

Kondacs and Watrous [KW97] introduced both 1-way and 2-way quantum finite automata (QFA), with emphasis on the more powerful 2-way automata. However, the model of 2-way QFA contradicts the idea of a system with small quantum mechanical part - it allows superpositions of the head of the QFA at different input locations; indeed, using such superpositions was the main idea in the proof [KW97] that 2-way QFA are more powerful than classical finite automata. Hence, to implement a 2-way QFA, we must represent the whole input in quantum form;

* Research supported by the Swedish Institute, Project ML-2000
** Research supported by Grant No.01.0354 from the Latvian Council of Science, by the Swedish Institute, Project ML-2000, and by the European Commission, Contract IST-1999-11234 (QAIP)

otherwise, the QFA would show to the environment which input symbol is being read, and its superposition would decohere. This may be very expensive, as the number of required quantum bits is linear in the size of input.

Hence, we think that more attention should be given to the study of 1-way QFA. This is a very reasonable model of computation and it is easy to see how it can be implemented. The finite dimensional state space of a QFA corresponds to a system with finitely many particles. Each letter has a corresponding unitary transformation on the state space. A classical device can read symbols from the input and apply the corresponding transformations to the quantum mechanical part. In fact, several practical experiments in quantum computing can be viewed as building such systems.

The results about 1-way QFA in [KW97] were quite pessimistic. It was shown that the class of languages recognized by 1-way QFA is a proper subset of regular languages. We continue to investigate 1-way QFA and show that, despite being very limited in some situations, they can perform quite well in other situations.

Our first results compare 1-way QFA and 1-way reversible automata. Clearly, the latter are a special case of the former; in particular, they cannot recognize all regular languages. The question then is whether 1-way QFA are more powerful than 1-way reversible automata. Interestingly, the answer depends on the accepting probability of a QFA. It was proved by Ambainis and Freivalds [AF98] that if a QFA gives correct answer with a large probability (greater than $\frac{7}{9}$), then the QFA can be replaced by a 1-way reversible deterministic finite automaton. However, this is not true for 0.68... and smaller probabilities. This effect was investigated in much detail by Ambainis and Ķikusts in [AK01]. It was found that if a QFA gives correct answer with a probability greater than $\frac{52+4\sqrt{7}}{81} = 0.7726...$, then the QFA can be replaced by a 1-way reversible deterministic finite automaton. On the other hand, there is a language which can be recognized by a QFA with probability $\frac{52+4\sqrt{7}}{81}$ but cannot be recognized by any reversible deterministic finite automaton.

It was shown by Ambainis and Freivalds that QFA can be much more space-efficient than deterministic and even probabilistic finite automata. Namely, to check whether the number of letters received from the input is divisible by a prime $p$, a 1-way QFA may need only $\log p$ states (this is equivalent to $\log \log p$ bits of memory), while any deterministic or probabilistic finite automaton needs $p$ states ($\log p$ bits of memory). We think that this space-efficient quantum algorithm may be interesting for designing other quantum algorithms as well.

## 2 Definitions and earlier results

Following [TB73], we define a probabilistic automaton as a triple $< Q, X, \pi >$, where $Q$ and $X$ are finite alphabets (states and input, respectively), and $\pi$ (the transition probability function) is a mapping on the product $X \times Q \times Q$ into the unit interval $[0, 1]$ such that $\sum_{q \in Q} \pi(a, q', q) = 1$ for all $q'$ in $Q$.

The construction of concrete languages for which the size of deterministic automata exceeds the size of all equivalent probabilistic automata has proved

to be quite a complicated problem. Freivalds [Fre82] constructed probabilistic finite automata with $n$ states for which the smallest equivalent deterministic automaton contains $\Omega(2^{\sqrt{n}})$ states. The problem of constructing probabilistic automata with $n$ states such that any equivalent deterministic automaton contains $a^n$ states still remains open.

Ambainis [Amb96] constructed a language accepted by a probabilistic automaton with $n$ states, and such that any deterministic automaton accepting this language has $\Omega(2^{n/\log n})$ states. He considered the language $L_m$ in an alphabet with $m$ letters, consisting of all words, which contain each letter of the alphabet exactly $m$ times.

Rasščevskis [Ras00] constructed a language accepted by a probabilistic automaton with $n$ states, and such that any deterministic automaton accepting this language has $2^n$ states. However, this result is not as strong as it may seem: while Ambainis used probabilistic automata with probability bounded away from $\frac{1}{2}$, Rasščevskis used probabilistic automata with unbounded probability.

In the present paper, we modify Rasščevskis' construction in order to get impressive size advantages of quantum finite automata over the deterministic ones. Since Rasščevskis' result [Ras00] was presented at a small conference in Sweden with limited distribution of proceedings, we include a section describing his construction.

We consider 1-way QFA as defined in [KW97]. Namely, a 1-way QFA is a tuple $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ where $Q$ is a finite set of states, $\Sigma$ is an input alphabet, $\delta$ is a transition function, $q_0 \in Q$ is a starting state and $Q_{acc} \subset Q$ and $Q_{rej} \subset Q$ are sets of accepting and rejecting states. The states in $Q_{acc}$ and $Q_{rej}$ are called *halting states* and the states in $Q_{non} = Q - (Q_{acc} \cup Q_{rej})$ are called *non-halting states*. Additional symbols # and %, not belonging to $\Sigma$, are used as left and right endmarker, respectively. The *working alphabet* of $M$ is $\Gamma = \Sigma \cup \{\#, \%\}$.

A *superposition* of $M$, generically denoted by the letter $\psi$, is any element of $l_2(Q)$, the linear space of all complex valued functions on $Q$. For $q \in Q$, we let $|q\rangle$ denote the unit vector with value 1 at $q$ and 0 elsewhere. All elements of $l_2(Q)$ can be expressed as linear combinations of the vectors $|q\rangle$.

The transition function $\delta$ maps $Q \times \Gamma \times Q$ into the complex field $\mathbb{C}$. The value $\delta(q, a, q')$ is the coefficient of $|q'\rangle$ in the superposition of states to which $M$ goes from $|q\rangle$ after reading the symbol $a$. For $a \in \Gamma$, we let $V_a$ denote the linear transformation on $l_2(Q)$ defined by

$$V_a |q\rangle = \sum_{q' \in Q} \delta(q, a, q') |q'\rangle.$$

We require all $V_a$ to be unitary.

The computation of a QFA starts in the superposition $|q_0\rangle$. Then transformations corresponding to the left endmarker #, the letters of the input word $x$, and the right endmarker %, are successively applied.

A transformation corresponding to a symbol $a \in \Gamma$ consists of two steps. First, $V_a$ is applied is applied to the current superposition $\psi$, resulting in a new

superposition $\psi' = V_a\psi$. Then, $\psi'$ is observed with respect to the observable $E_{acc} \otimes E_{rej} \otimes E_{non}$, where $E_i$ is the linear subspace of $l_2(Q)$ spanned by $|q\rangle$ for $q \in Q_i$, $i \in \{acc, rej, non\}$. This observation gives $x \in Q_i$ with probability equal to the amplitude of the orthogonal projection $P_i\psi'$ of $\psi'$ onto $E_i$. As a result of the observation, the superposition $\psi'$ collapses to one of the projections $P_i\psi'$, $i \in \{acc, rej, non\}$; if $i = acc$, the input $x$ is accepted; if $i = rej$, the input $x$ is rejected; if $i = non$, the transformation corresponding to the next letter in $x$ is applied to $P_{non}\psi'$.

We regard the composition $P_iV_a\psi$ as reading a letter $a$ in superposition $\psi$. We put $V'_a = P_{non}V_a$, and for a word $x = a_1 \dots a_k$, we put $V_x = V_{a_k} \dots V_{a_1}$ and $V'_x = V'_{a_k} \dots V'_{a_1}$. Further, we let $\psi_x$ denote the non-halting part of the QFA's configuration after reading the word $x$. It is easy to see that for any word $x$ and letter $a$, $\psi_{xa} = V'_a(\psi_x)$.

Ambainis and Freivalds [AF98] considered, for prime $p$, the languages $L_p$ in a single letter consisting of all words of length divisible by $p$.

**Theorem 1.** *For any $\epsilon > 0$, there is a QFA with $O(\log p)$ states recognizing $L_p$ with probability $1 - \epsilon$.*

**Theorem 2.** *Any deterministic 1-way finite automaton recognizing $L_p$ has at least $p$ states.*

The number of states needed by a 1-way probabilistic finite automaton to recognize a language is often close to the logarithm of the number of states needed by a deterministic automaton [Amb96,Fre82]. However, this is not the case with $L_p$.

**Theorem 3.** [AF98] *Any 1-way probabilistic finite automaton recognizing $L_p$ with probability $1/2 + \epsilon$, for a fixed $\epsilon > 0$, has at least $p$ states.*

**Corollary 1.** [AF98] *There are languages, for the recognition of which the number of states needed by a probabilistic automaton is exponential in the number of states needed by a 1-way QFA.*

## 3 Probabilistic vs. deterministic automata

In this section, we describe Rasščevskis' result [Ras00]. Consider the language $L_n$ in the alphabet $\{a, b\}$ of two letters, consisting of all words of length at least $n$, which have $a$ as the $n$-th letter from the end.

**Lemma 1.** *Any deterministic finite automaton recognizing $L_n$ has at least $2^n$ states.*

*Proof.* There are exactly $2^n$ words of length $n$. Any two of these differ on at least one letter; suppose the first such letter is the $m$-th one. Now, if we add sequence of $m - 1$ letters to each of the two words, then one of the new words should be accepted but the other rejected. Any deterministic finite automaton recognizing $L_n$ must therefore remember any of the $2^n$ words differently, thus having at least $2^n$ states. □

Consider now, for any natural $n$, a probabilistic finite automaton $U = U_n$ with initial state $S$, accepting states $A$ and $Q_n$, and working states $Q_0$,$Q_1$, ..., $Q_{n-1}$. From $S$, without reading any input, $U$ passes to $A$ with probability $\frac{1}{2} - \delta$, and to $Q_0$ with probability $\frac{1}{2} + \delta$; the variable $\delta$ is actually the radius of isolation for $U$, and for $U$ to work correctly, we will need $\delta \leq \frac{1}{4e(n+1)}$. From $A$, $U$ passes to $A$ on any input. From $Q_0$, on input $a$, $U$ passes to $Q_0$ with probability $x$, and it passes to $Q_1$ with probability $1 - x$ (the value of $x$ is for now free to choose, but the optimal value will turn out to be $\frac{n}{n+1}$); on input $b$, the automaton stays in $Q_0$. From $Q_i$, $1 \leq i < n$, $U$ passes to $Q_{i+1}$, and from $Q_n$, it passes to $Q_0$.

**Lemma 2.** *$U$ rejects words not in $L_n$ with probability $\frac{1}{2} - \delta$.*

*Proof.* For the accepting states $A$ and $Q_n$, the probability of $A$ is always $\frac{1}{2} - \delta$ and the probability of $Q_n$ equals the probability of state $Q_1$ of $n-1$ inputs ago. But, as the probability of $Q_1$ is 0 if last input is $b$ (if the last input was $a$, the $n$-th letter from the end of the word $l$ would be $a$, contradicting $l \notin L_n$), the probability of $Q_n$ is always zero. □

**Lemma 3.** *The probability in state $Q_0$ is never less than $(\frac{1}{2} + \delta)x^n$.*

*Proof.* Assume the input word has less than $n$ letters. At the beginning probability at the state $Q_0$ is and after each letter the input automaton passes from $Q_0$ to $Q_0$ with probability at least $x$. Hence the probability of $Q_0$ is at least $(\frac{1}{2} + \delta)x^n$ and as $k < n$, it is more than $(\frac{1}{2} + \delta)x^n$.

Assume now the input word has $n$ or more letters. Denote by $p_i$ the probability of the state $Q_i$, $0 \leq i \leq n$, for this word with the last $n$ letters removed (in other words, the same word $n$ inputs ago). The probability of the state $Q_0$ now is then obviously bounded from below by the sum $p(x) = p_0 x^n + \sum_{1 \leq i \leq n} p_i x^{i-1}$. The bound is attained if the last $n$ input letters are all $a$; in any other case $Q_0$ passes to $Q_0$ with probability 1 instead of $x$, but

$$p(x) \geq (p_0 + p_1 + \cdots + p_n)x^n = (\frac{1}{2} + \delta)x^n.$$

□

**Lemma 4.** *$U$ accepts words in $L_n$ with probability at least $\frac{1}{2} - \delta$.*

*Proof.* Suppose that $n$ inputs ago the probability of the state $Q_0$ was $p$. If $l \in L_n$, the probability of $Q_1$ of $n-1$ inputs ago was $p(1-x)$. But the probability of $Q_n$ equals the probability of $Q_1$ of $n-1$ inputs ago. Hence it equals $p(1-x)$. As $Q_n$ and $A$ are accepting states and the probability of $A$ is always $\frac{1}{2} - \delta$ and $p > (\frac{1}{2} - \delta)x^n$, the automaton $U$ accepts the word $l$ with probability at least

$$\frac{1}{2} - \delta + p(1-x) \geq \frac{1}{2} - \delta + (\frac{1}{2} + \delta)x^n(1-x).$$

For this number to be grater than $\frac{1}{2} + \delta$, we need $(\frac{1}{2} + \delta)x^n(1-x) > 2\delta$; it would be enough if $x^n(1-x) > 4\delta$. For fixed $n$, the last left-hand expression attains its maximum in the interval $[0, 1]$ at $x = \frac{n}{n+1}$. For this value of $x$, we may take any $\delta$ less than $\frac{1}{4e(n+1)} < \frac{1}{4}(\frac{n}{n+1})^n \frac{1}{n+1}$. □

**Theorem 4.** *There exists a probabilistic finite automaton with $n$ states, isolated cutpoint and radius of isolation $\Omega(n^{-1})$ such that the smallest equivalent deterministic finite automaton contains $\Omega(2^n)$ states.*

*Proof.* Consider the automaton $U$ with $n$ states (it accepts the language $L_{n-3}$). Then any equivalent deterministic automaton would have at least $2^{n-3}$ states but the probabilistic automaton constructed above works correctly with radius of isolation equal to $\frac{1}{4e(n-2)}$. □

## 4 Quantum vs. deterministic automata

Consider the language $L$ consisting of words $x_1x_2x_3\dots x_l$ in the alphabet $\{a,b\}$ such that among the symbols $x_{l-n+1}, x_{l-2n+1}, x_{l-3n+1}, \dots, x_{l-mn+1}$ there is an odd number of symbols $a$, where $m$ is an integer such that $l-mn+1>0$ and $l-(m+1)n+1\leq 0$.

**Lemma 5.** *Any deterministic finite automaton recognizing the language $L$ has at least $2^n$ states.*

*Proof.* For arbitrary symbol $x_k$, we describe the set $x_{k-2n+1}, x_{k-n+1}, x_{k-2n+1}$, $x_{k-3n+1}, \dots, x_{k-mn+1}$ as the *predecessors* of the symbol $x_k$. For every symbol in $\{x_l, x_{l-1}, x_{l-2}, \dots, x_{l-n+1}\}$ the automaton is to remember whether or not there is an odd numbers of symbols $a$ in the set of the predecessors. □

**Lemma 6.** *There is a quantum finite automaton with $2n+1$ non-halting states and $2n+1$ halting states recognizing the language $L$ with a bounded error.*

*Proof.* Let $q_0$ be the initial state, and let $q_1, q_2, q_3, \dots, q_{2n}$ be the remaining non-halting states. When the start-marker comes in, the computation is directed to the states $q_1, q_2, q_3, \dots, q_n$ with equal amplitudes $(\frac{n^2-2n}{4n^2-8n+4})^{\frac{1}{2}}$ each, and with the amplitude $(\frac{n-2}{2n-2})^{\frac{1}{2}}$ to an accepting state.

When the symbol $b$ comes in, the states $q_1, q_2, q_3, \dots, q_n$ are changed cyclically, and so are the states $q_{n+1}, q_{n+2}, q_{n+3}, \dots, q_{2n}$.

When the symbol $a$ comes in, the same transformation is performed with one exception: $q_1$ is followed by $q_{n+2}$, and $q_{n+1}$ is followed by $q_2$.

When the end-marker comes in, the state $q_{n+1}$ is followed by the only accepting state, and all the remaining states $q_1, q_2, q_3, \dots, q_n, q_{n+2}, q_{n+3}, \dots, q_{2n}$ are followed by $2n-1$ different rejecting states, respectively.

If the input word is in the language, the probability to accept it at the very first step is $\frac{n-2}{2n-2}$, and the probability to accept it after reading the end-marker is $(\frac{n^2-2n}{4n^2-8n+4})^{\frac{1}{2}}$. If the input word is not in the language, the probability to reject it is $(n-1)\frac{n^2-2n}{4n^2-8n+4} = \frac{n-2}{2n-2}$. □

This results in our main theorem.

**Theorem 5.** *For any natural $n$ there is a language recognized with bounded error by some quantum finite automaton with $4n+2$ states, but not recognized by any deterministic finite automaton with less than $2^n$ states.*

## References

[Amb96] A. Ambainis. The complexity of probabilistic versus deterministic finite automata. *Proc. Int. Symp. on Algorithms and Computation (ISAAC'96), Lecture Notes in Computer Science*, vol. 1178, 1996, pp. 233-239.

[AF98] A. Ambainis and R. Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. *Proc. 39th IEEE Conf. on Foundations of Computer Science*, 1998, pp. 332-341.

[AK01] A. Ambainis and A. Ķikusts. Exact results for accepting probabilities of quantum automata. *Lecture Notes in Computer Science*, vol. 2136, 2001, pp. 135-147.

[Fre82] R. Freivalds. On the growth of the number of states in result of determinization of probabilistic finite automata. *Avtomatika i Vychislitel'naya Tehnika*, No. 3, 1982, pp. 39-42. (in Russian)

[KW97] A. Kondacs and J. Watrous. On the power of quantum finite state automata. *Proc. 38th IEEE Conf. on Foundations of Computer Science,* 1997, pp. 66-75.

[MC00] C. Moore and J. P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, vol. 237, 2000, pp. 275-306.

[Pin87] J. Pin. On the languages accepted by finite reversible automata. *Lecture Notes in Computer Science*, vol. 267, 1987, pp. 237-249.

[Rab63] M. Rabin. Probabilistic automata. *Information and Control*, vol. 6, 1963, pp. 230-245.

[Ras00] Z. Rasščevskis. The complexity of probabilistic versus deterministic finite automata. In: R. Bonner and R. Freivalds (Eds.), *Proc. Int. Workshop on Quantum Computation and Learning, Sundbyholm, 27-29 May 2000*, Institute of Mathematics and Computer Science, University of Latvia, and Department of Mathematics and Physics, Mälardalen University, Sweden, 2000, pp. 89-92.

[TB73] B. A. Trakhtenbrot and J. Bārzdiņš. *Finite Automata: Behavior and Synthesis,* North-Holland, 1973.

# Computation of Boolean Functions by Probabilistic and Quantum Decision Trees

Vasilijs Kravcevs*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
sd80004@lanet.lv

**Abstract.** We consider the computation of total Boolean functions by deterministic, probabilistic, and quantum decision trees. The complexity of a Boolean function is in each case defined in terms of the least number of queries to an oracle, required for its computation. We exhibit some Boolean functions that can be computed by probabilistic and quantum decision trees with high probability of correct answer, and use fewer queries than deterministic decision trees.

## 1 Introduction

We consider the computation a Boolean function $f$ of a vector $x = (x_1, \ldots, x_N)$ of binary variables, with $x$ treated as a black-box (or oracle); here and henceforth we only consider total functions with no fictive variables. Calling the oracle on $i$ returns the value of $x_i$; such a call is called a query. We measure the complexity of a algorithm for computing $f$ by the number of queries it makes; clearly, the aim is to make as few queries as possible.

A classical deterministic algorithm for computing $f$ by oracle queries is often called a *decision tree*, as it can be represented by a binary tree with queries as nodes, each node branching out by the outcome of the query (0 or 1), the leaves of the tree representing the function values, $f(x) = 0$ or $f(x) = 1$. The complexity (or cost) of such an algorithm is the number of queries made in the worst case of $x$; this is the depth of the tree. By *decision tree complexity* of $f$, denoted $D(f)$, we understand the cost of the shortest decision tree, which computes $f$.

Computing $f$ by oracle queries may be randomized by choosing queries with probabilities conditional on the outcome of a previous query. We may then define the *probabilistic* decision tree complexity of $f$, generically denoted $R(f)$, as the minimum of the expected numbers of queries over all probabilistic decision trees computing $f$, in the worst case of $x$; here, we have the choice of computing $f$ *exactly*, or with *bounded error*.

A *quantum* decision tree (or quantum network) is the quantum analogue of a classical decision tree, where queries and other operations are made in quantum

* Research supported by contract IST–1999–11234 (QAIP) from the European Commision and grant no. 01.0354 from the Latvian Council of Science.

superposition. Such a network may be represented as a sequence of unitary transformations; the quantum decision tree complexity of $f$ is then defined in terms of the least number of such transformations.

A natural task is to compare deterministic, probabilistic, and quantum decision tree complexities of a Boolean function. Consider, for example, the function $\mathrm{OR}(x) = x_1 \vee \ldots \vee x_N$. It is well known that the least number of queries for computing OR by classical deterministic algorithms is $N$, and for probabilistic algorithms, it is $\Theta(N)$. Grover [8] discovered a quantum algorithm, which allows computing OR with small error probability with only $\Theta(\sqrt{N})$ queries, and it was shown in [3, 4, 11] that this number is asymptotically optimal. For total Boolean functions, the Grover quadratic speed-up is also the best quantum speed-up known; see [10] for a discussion of the simple Boolean functions OR, AND, XOR.

In this paper, we consider more complex Boolean functions, and construct probabilistic and quantum decision trees for their computation, the latter with a quantum speed-up.

## 2 Definitions

We adopt the usual set-up [10, 8, 50, 10]. Given a vector $x = (x_1, \ldots, x_N)$ of $N$ Boolean variables and a Boolean function $f : \{0,1\}^N \to \{0,1\}$, the problem is to compute $f(x)$ by calling an oracle; classically, an oracle receives an index $i$ and outputs the value of the $i$-th variable $x_i$; such a call to the oracle is called a query. We consider three models of computation of a Boolean function.

A *deterministic decision tree* for $f$ is a rooted directed binary tree, in which the nodes are queries, and the leaves, labelled 0 or 1, represent the outcomes $f(x)$ of the computation. Each query has two outgoing edges, labelled by its outcome. A path from the root to a leaf of the tree is a sequence of queries, the choice of a subsequent query determined by the outcome of the previous one.

A *probabilistic decision tree* for $f$ is a rooted directed tree (not necessarily binary), in which each inner node is a query, and the leaves, labelled 0 or 1, represent the outcomes of computation. The edges leaving the root are labelled with the (non-zero) probabilities of choosing their target queries at the first step of computation. The edges leaving a query have double labels $(\xi, p)$, the first representing the outcome of the query, the second being the (non-zero) probability of choosing the target query of the edge; the probabilities of all the edges leaving a node sum up to one.

A *quantum decision tree* (or *quantum network*) is a quantum analogue of a probabilistic decision tree (every probabilistic decision tree may be represented as a quantum decision tree, but not conversely), where queries and other operations are made in quantum superposition. Such a decision tree may be represented as finite sequence $U_1$, $O_1$, $U_2$, $O_2$, ..., where $U_i$ are arbitrary unitary transformations, and $O_j$ are unitary transformations corresponding to the queries to the oracle. The computation ends with a measurement of the final state. The measurement gives the probability of receiving the value of the function $f(x)$.

A randomized algorithm *exactly computes* $f(x)$ if on any input $x$ it outputs $f(x)$ with probability 1. A randomized algorithm computes $f(x)$ with *bounded error* if it outputs $f(x)$ with probability at least $\frac{1}{2}+\epsilon$ for some $\epsilon > 0$. By $D(f)$, $P_E(f)$, $P_2(f)$, $Q_E(f)$, $Q_2(f)$, we denote the least number of queries required by deterministic, exact probabilistic, probabilistic with bounded-error, quantum exact, quantum bounded-error algorithms, respectively. For example, for $\mathrm{OR}(x) = x_1 \vee \ldots \vee x_N$ we have $D(f) = N$, $P_2(f) \in \Theta(N)$, and $Q_2(f) \in \Theta(\sqrt{N})$. Note that the complexity of a Boolean function measured by the number of queries does not capture the complexity of any auxiliary computational steps normally performed in the computation.

The *Hamming weight* of $x$, denoted by $|x|$, is the number of 1:s in $x$. A Boolean function $f$ is *symmetric* if permuting the input does not change the function value (i.e. $f(x)$ depends on $|x|$ only). For such $f$ let $f_k = f(x)$ for $|x| = k$; and define

$$\Gamma(f) = \min\{|2k - N + 1| : \ f_k \neq f_{k+1},\, 0 \leq k < N\}.$$

Roughly, $\Gamma(f)$ is low if $f_k$ 'jumps near the middle'. Recall:

**Theorem 1.** [10] *If $f(x_1, \ldots, x_N)$ is non-constant and symmetric, then $Q_2(f) \in \Theta(\sqrt{N(N - \Gamma(f))})$.*

For the sequel, an obvious but clarifying remark may be in order. A Boolean function $f$ of $N$ binary variables is nothing but a dichotomy of $\{0,1\}^N$, and hence the computation of $f$ is nothing but the recognition of the finite language $f^{-1}(1)$. We may thus use the language of automata, in which a word $x$ is accepted or rejected if and only if $f(x) = 1$ or $f(x) = 0$, respectively.

## 3 Some Boolean functions and their decision trees

### 3.1 The function AND

To begin with, consider the function $\mathrm{AND}(x_1, x_2) = x_1 \wedge x_2$. There is a deterministic decision tree for AND containing two queries: we ask about the first variable, and, if the answer is 1, we make a query about the second variable.

A probabilistic decision tree computing AND with bounded error $\frac{1}{3}$ is obtained as follows; we use the language of automata in the description. From the root, we go to the rejecting state (0-leaf) with probability $\frac{1}{3}$, and with equal probabilities $\frac{1}{3}$ we pass to the two nodes querying the oracle about $x_k$, $k = 1, 2$. If the answer of a query is 1, we go to the accepting state (1-leaf) with probability 1; if the answer is 0, we go to the rejecting state with probability 1. The results are summarized in Table 1: we accept in case $x_1 = x_2 = 1$ with probability $\frac{2}{3}$; we reject in case $x_1 = x_2 = 0$ with probability 1; and we reject in all other cases with probability $\frac{2}{3}$.

| $x$ | $y$ | $p_1$ | $p_0$ |
|---|---|---|---|
| 00 | 0 | 0 | 1 |
| 01 | 0 | $\frac{1}{3}$ | $\frac{2}{3}$ |
| 10 | 0 | $\frac{1}{3}$ | $\frac{2}{3}$ |
| 11 | 1 | $\frac{2}{3}$ | $\frac{1}{3}$ |

Table 1. Probabilities $p_k = p_k(x)$ of output $k$, $k = 0, 1$, for our probabilistic decision tree computing the function $y = \mathrm{AND}(x)$.

So, we obtain $P_2(\mathrm{AND}) = 1$ with probability $\frac{2}{3}$ of correct answer. A quantum decision tree, which we are about to describe, also gives $Q_2(\mathrm{AND}) = 1$, but with a better probability. It works as follows. From the root, we go to a state $q_3$ with amplitude $\frac{1}{\sqrt{3}}$, and query $x_1$ and $x_2$ with equal amplitudes $\frac{1}{\sqrt{3}}$. For each of the two queries $x_k$, if the answer to $x_k$ is 0, we go to a state $q_k$ with conditional amplitude $-1$; if the answer is 1, we go to the state $q_k$ with conditional amplitude $+1$. Next, we perform the Hadamard transform

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{1}$$

on the states $q_1$ and $q_2$, and receive states $q_4, q_5$. Then, we perform the Hadamard transform on the states $q_4$ and $q_3$, the first outcome state being accepting one, but the second outcome state and state $q_5$ being rejecting states. The results are summarized in Table 2: we accept in the case $x_1 = x_2 = 1$ with probability $\frac{6+\sqrt{2}}{12}$, we reject in the case $x_1 = x_2 = 0$ with the same probability, and we reject in all other cases with probability $\frac{5}{6}$.

| $x$ | $y$ | $A(q_5)$ | $A(q_6)$ | $A(q_7)$ | $p_1$ | $p_0$ |
|---|---|---|---|---|---|---|
| 00 | 0 | 0 | $\frac{-2+\sqrt{2}}{\sqrt{12}}$ | $\frac{-2-\sqrt{2}}{\sqrt{12}}$ | $\frac{6-4\sqrt{2}}{12}$ | $\frac{6+4\sqrt{2}}{12}$ |
| 01 | 0 | $-\frac{2}{\sqrt{6}}$ | $\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $\frac{1}{6}$ | $\frac{5}{6}$ |
| 10 | 0 | $\frac{2}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $\frac{1}{6}$ | $\frac{5}{6}$ |
| 11 | 1 | 0 | $\frac{2+\sqrt{2}}{\sqrt{12}}$ | $\frac{2-\sqrt{2}}{\sqrt{12}}$ | $\frac{6+4\sqrt{2}}{12}$ | $\frac{6-4\sqrt{2}}{12}$ |

Table 2. Probabilities $p_k = p_k(x)$ of output $k$, $k = 0, 1$, and amplitudes $A(q_k)$ for our quantum decision tree computing the function $y = \mathrm{AND}(x)$.

We conclude that quantum decision trees may work better than probabilistic decision trees.

### 3.2 The function $v$

Consider the function $v(x) = (x_1, x_2, x_3)$ which is equal to one if and only if the count of 1:s in $x$ is at least two. It is easy to see that $D(v) = 3$.

**Theorem 2.** *There is a probabilistic decision tree with two queries computing $v$ with probability $\frac{3}{5}$.*

*Proof.* We go to the accepting state with probability $\frac{2}{5}$ and with probabilities $\frac{1}{5}$ we query the oracle about two variables $x_k$ and $x_{(k+1) \bmod 3}$, $k = 1, 2, 3$; if both variables are 1 then we go to the accepting state with local probability 1, otherwise we go to the rejecting state with local probability $\frac{1}{5}$. We accept in the cases $x_1 = x_2 = x_3$ with probability $\frac{3}{5}$. We reject in all other cases with probability $\frac{3}{5}$. □

**Theorem 3.** *There is a quantum decision tree with two queries computing $v$ with probability $\frac{2}{3} + \epsilon$.*

*Proof.* We go to the state $q_0$ with amplitude $\frac{1}{2}$ and query oracle about $x_1$ and $x_2$, $x_1$ and $x_3$, $x_2$ and $x_3$, with amplitudes $\frac{1}{2}e^0$, $\frac{1}{2}e^{\frac{2\pi i}{3}}$, $\frac{1}{2}e^{\frac{4\pi i}{3}}$, respectively. If both variables are 1, we go to states $q_1$, $q_2$, $q_3$, respectively, with conditional amplitude $+1$; if not, we to the state $q_k$ with conditional amplitude $-1$. Next, we perform quantum Fourier transform

$$\frac{1}{\sqrt{3}} \begin{pmatrix} e^0 & e^0 & e^0 \\ e^0 & e^{\frac{2\pi i}{3}} & e^{\frac{4\pi i}{3}} \\ e^0 & e^{\frac{4\pi i}{3}} & e^{\frac{2\pi i}{3}} \end{pmatrix} \tag{2}$$

on the states $q_1, q_2, q_3$, receiving states $q_4, q_5, q_6$. Then, we perform the Hadamard transform on the states $q_0$ and $q_6$. The first outcome state and states $q_4, q_5$ are accepting, the second outcome state is rejecting.

The results are summarized in Table 3. We reject in all cases when at least two variables are 0 with probability $(\frac{1+\sqrt{3}}{2\sqrt{2}})^2 = \frac{4+2\sqrt{3}}{8} > \frac{7}{8}$. We accept in cases when exactly two variables are 1 with probability greater than $\frac{2}{3}$, and we accept when all the variables are 1 with probability greater than $\frac{7}{8}$. □

| $x$ | $A(q_4)$ | $A(q_5)$ | $A(q_7)$ | $A(q_4)$ | $p_1$ | $p_0$ |
|---|---|---|---|---|---|---|
| $\|x\| = 1$ | 0 | 0 | $\frac{1-\sqrt{3}}{2\sqrt{2}}$ | $\frac{1+\sqrt{3}}{2\sqrt{2}}$ | $< \frac{1}{8}$ | $> \frac{7}{8}$ |
| 011 | $\frac{-1}{2\sqrt{3}} - \frac{1}{2}i$ | $\frac{-1}{2\sqrt{3}} + \frac{1}{2}i$ | $\frac{-1+\sqrt{3}}{2\sqrt{6}}$ | $\frac{1+\sqrt{3}}{2\sqrt{6}}$ | $> \frac{2}{3}$ | $< \frac{1}{3}$ |
| 101 | $\frac{-1}{2\sqrt{3}} + \frac{1}{2}i$ | $\frac{-1}{2\sqrt{3}} - \frac{1}{2}i$ | $\frac{-1+\sqrt{3}}{2\sqrt{6}}$ | $\frac{1+\sqrt{3}}{2\sqrt{6}}$ | $> \frac{2}{3}$ | $< \frac{1}{3}$ |
| 110 | $\frac{1}{\sqrt{3}}$ | $\frac{1}{\sqrt{3}}$ | $\frac{-1+\sqrt{3}}{2\sqrt{6}}$ | $\frac{1+\sqrt{3}}{2\sqrt{6}}$ | $> \frac{2}{3}$ | $< \frac{1}{3}$ |
| 111 | 0 | 0 | $\frac{1+\sqrt{3}}{2\sqrt{2}}$ | $\frac{1-\sqrt{3}}{2\sqrt{2}}$ | $> \frac{7}{8}$ | $< \frac{1}{8}$ |

Table 3. Probabilities $p_k = p_k(x)$ of output $k$, $k = 0, 1$, and amplitudes $A(q_k)$ for our quantum decision tree computing the function $y = \mathrm{AND}(x)$.

### 3.3 The function $V$

Consider now the function $V(x) = (x_1, \ldots, x_N)$, $N = 2p + 1$, which is equal to one if and only if the count of 1:s in $x$ is at least $p + 1$. We get no remarkable speed-up.

**Theorem 4.** $Q_2(V) \in \Theta(N)$.

*Proof.* The function $V(x) = (x_1, \ldots, x_N)$ is symmetric. Invoke Theorem 1, observing that $v_k \neq v_{k+1}$ only when $k = (N-1)/2$, and hence $\Gamma(V) = 0$. □

### 3.4 The function $J_N$

Put $J_N(x_1, \ldots, x_N) = (x_1 \wedge \ldots \wedge x_N) \vee (\bar{x}_1 \wedge \ldots \wedge \bar{x}_N)$. Freivalds and Lace [7] gave a quantum decision tree with one query computing $J_3$ with probability $\frac{2}{3}$. We improve this:

**Theorem 5.** *There is a quantum decision tree with one query computing $J_3$ with probability $\frac{8}{9}$.*

*Proof.* We query $x_k$, $k = 1, 2, 3$, with amplitudes $\frac{1}{\sqrt{3}}$; if the answer is 0, we go to the state $q_k$ with conditional amplitude $-1$; if the answer is 1, we go to the state $q_k$ with conditional amplitude $+1$. Next, we perform the quantum Fourier transform (2) on the states $q_1$, $q_2$, $q_3$. The first outcome state is accepting, the other two states are rejecting. We accept if $x_1 = x_2 = x_3$ with probability 1, we reject in all other cases with probability $\frac{8}{9}$. □

**Theorem 6.** $Q_2(J_N) \in \Theta(\sqrt{N})$.

*Proof.* The function $f = J_N$ is symmetric. Invoke Theorem 1, observing that $\Gamma(f) = N - 1$ since $f_k \neq f_{k+1}$ only when $k = 0$ or $k = N - 1$, and in both cases $|2k - N + 1| = N - 1$. □

## References


1. A. Ambainis and R. Freivalds. *1-way Quantum finite automata: strengths, weaknesses and generalizations.* Proc. FOCS'98, pp. 332-341. Also http://xxx.lanl.gov/abs/quant-ph/9802062.
2. R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. *Quantum lower bounds by polynomials.* Proc. FOCS'98, pp. 351-361. Also arXiv:quant-ph/9802049v3.
3. C. Bennett, E. Bernstein, C. Brassard and U. Vasirani. *Strengths and weaknesses of quantum computing.* SIAM Journal on Computing, 26(3), 1997, pp. 1510-1523. Also quant-ph/9701001.
4. M. Boyer, G. Brassard, P. Hoyer and A. Tapp. *Tight bounds on quantum searching.* Fortschritte der Physik, 46(5), 1998, pp. 493-505.
5. H. Buhrman, R. Cleve, R. Wolf and C. Zalka. *Bounds for small-error and zero-error quantum algorithms.* 1998, http://xxx.lanl.gov/abs/quant-ph/9802062.
6. E. Farhi, J. Coldstone, S. Gutmann and M. Sipser. A limit on the speed of quantum computation determining parity. 1998. http://xxx.lanl.gov/abs/quant-ph/9802045.
7. R. Freivalds and L. Lace. *Better upper bounds for advantages of probabilistic and quantum decision trees.* Unpublished manuscript, 2002.
8. L. K. Grover. *A fast quantum mechanical algorithm for database search.* Proc. 28th STOC, 1996, pp. 212-219.
9. J. Gruska. *Quantum computing,* McGraw Hill, 1999.
10. P. Hajnal. *Decision tree complexity of Boolean functions.* Coll. Math. Soc. Janos Bolyai 60, Sets, graphs and numbers, Budapest, 1991.
11. N. Nisan and M. Szedegy. *On the degree of Boolean functions as real polynomials.* Computational Complexity, 4(4), 1994, pp. 301-313.
12. C. Zalka. *Grover's quantum searching algorithm is optimal.* 1997. http://xxx.lanl.gov/abs/quant-ph/9711070.

# Some Examples to Show the Advantages of Probabilistic and Quantum Decision Trees

Lelde Lāce*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
lelde@mii.lu.lv

**Abstract.** Decision trees (query algorithms) are tools for computing Boolean functions by asking for the values of Boolean variables. A natural complexity measure is the number of queries. It is well known that some Boolean functions may be computed by probabilistic and quantum decision trees with fewer queries than required by deterministic decision trees. We examine this phenomenon in some examples.

## 1 Introduction

Recently it has become clear that a quantum computer could in principle solve certain problems faster than a conventional computer. A quantum computer is a device which takes full advantage of quantum mechanical superposition and interference. Building an actual quantum computer is probably far off in the future.

The Boolean decision trees model is the simplest model of computation of Boolean functions. In this model, the primitive operation made by an algorithm is the evaluation of a Boolean input variable. The cost of a (deterministic) algorithm is the number of variables it evaluates in a worst case input. It is easy to find the deterministic complexity of all explicit Boolean functions (for most functions it is equal to the number of variables).

The *black-box* model of computation arises when one is given a black-box containing an $N$-tuple of Boolean variables $x = (x_1, x_2, ..., x_n)$. The box supplies output $x_i$ on input $i$. We wish to determine a property of $x$, i.e. a function $f : \{0,1\}^n \to \{0,1\}$, accessing the $x_i$:s only through the black-box. Such a black-box access is called a *query*. We want to compute such properties using as few queries as possible.

For example, to determine whether or not $x$ contains at least one 1, we compute the property $\mathrm{OR}(x) = x_1 \vee ... \vee x_n$. It is well known that the number of queries required to compute OR by any *classical* (deterministic or probabilistic) algorithm is $\Theta(n)$. Grover [7] discovered a remarkable *quantum* algorithm, which

* Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission, and the Swedish Institute, Project ML2000

makes queries in superposition, and can be used to compute OR with small error probability using only $O(\sqrt{n})$ queries.

In this paper, we give probabilistic decision trees for $\mathrm{OR}(x)$ and $\mathrm{AND}(x)$ with isolated cut point using only one query. In the quantum case, we give three quantum decision trees, which are all better than their deterministic counterpart.

## 2 Definitions

We follow closely Hajnal [9] and Vereshchagin [12].

**Definition 1.** *A (deterministic)* decision tree *is a binary tree with labels on each node and edge. Each inner node is labelled by a query. One of the two edges leaving the node is labelled 0, the other is labelled 1. The two labels represent the two possible answers to the query. The two subtrees at a node describe how the algorithm proceeds after receiving the corresponding answer. Each leaf is labelled 0 or 1. These labels give the output, i.e. the value of the function.*

Clearly, each truth-assignment to the variables determines a unique path, the *computation path*, from the root to a leaf of the tree. The Boolean function computed by the decision tree takes the label at this leaf as its value on the given input.

**Definition 2.** *Let* $\mathrm{cost}(A, x)$ *be the number of queries asked when the decision tree* $A$ *is executed on input* $x$*. This is the length of the computation path forced by* $x$*. The worst case complexity of* $A$ *is* $\max_x \mathrm{cost}(A, x)$*; it is the depth of the tree. The* decision tree complexity *of a Boolean function* $f$ *is*

$$C(f) = \min_A \max_x \mathrm{cost}(A, x),$$

*where the minimum is taken over all decision trees* $A$ *computing the function* $f$*.*

So the cost of a computation is just the number of queries asked. We ignore the time needed for the generation of queries and the computation of the output.

A *randomized decision tree* is a rooted, not necessarily binary, tree. Each of its inner nodes is labelled by a variable, i.e. by a query. The edges leaving a node are labelled 0 or 1. The subtrees which can be reached from a given node by an edge labelled 0 are the possible continuations of the algorithm after receiving the answer 0. The role of the edges labelled 1 is symmetric. During the execution of the algorithm the next step is chosen randomly.

We use the simplest convention when we require that the algorithm always gives the correct answer. Using the second formalization of the randomized decision tree, it computes a function $f$ if the distribution is non-zero only on deterministic trees computing $f$.

**Definition 3.** *Let* $A_1, \ldots, A_N$ *be the set of all deterministic decision trees computing the function* $f$*. Let* $R = p_1, \ldots p_N$ *be a randomized decision tree computing*

*$f$, where $p_i$ is the probability of $A_i$. The cost of $R$ on input $x$ is* $\mathrm{cost}(R,x) = \Sigma_i p_i \mathit{cost}(A_i,x)$. *The* randomized decision tree complexity *of a function $f$ is*

$$C^R(f) = \min_R \max_x (R,x),$$

*where the minimum is taken over all randomized decision trees computing the function $f$.*

For the basic notions of Quantum Computation we refer the monographs by Gruska [8], Nielsen and Chuang [11], and Hirvensalo [10]. A quantum computer performs a sequence of unitary transformations $U_1$, $U_2$, ..., $U_T$ on a complex Hilbert space, called the *state space*. The state space has a canonical orthonormal basis which is indexed by the configurations $s$ of some classical computer $M$. The basis state corresponding to $s$ is denoted by $|s\rangle$. The initial state $\phi_0$ is a basis state. At any point in time $t$, $1 \leq t \leq T$, the state $\phi_t$ is obtained by applying $U_t$ to $\phi_{t-1}$. At time $T$, we *measure* the state $\phi_T$.

We define a *quantum decision tree* following Deutsch and Jozsa [4] and Buhrman *et al* [3]. For input length $n$, the initial state $\phi_0$ is independent of the input $x = x_0 x_1 \ldots x_n$. We allow arbitrary unitary transformations independent of $x$. In addition, we allow $A$ to make *quantum queries*. These are transformation $U_x$ taking the basis state $|i,b,z\rangle$ to $|i,b \oplus x_i,z\rangle$, where:

- $i$ is a binary string of length $\log n$ denoting an index in the input $x$,
- $b$ is the contents of the location where the result of the oracle query will be placed,
- $z$ is a placeholder for the remainder of the state description,

and the comma denotes concatenation.

We define the *cost* of $A$ to be the largest number of times a query transformation $U_x$ is performed; all other transformations are free.

We want to compute a function $f : \{0,1\}^n \to \{0,1\}$ using as few queries as possible (on the worst-case input). We distinguish between three different error models. In the case of *exact* computation, an algorithm must always give the correct answer $f(x)$ for every $x$. In case of *bounded-error* computation, an algorithm must give the correct answer $f(x)$ with probability $\geq \frac{2}{3}$ for every $x$. In the case of *zero-error* computation, an algorithm is allowed to give the answer 'don't know' with probability $\leq \frac{1}{2}$, but if outputs an answer (0 or 1), then this must be the correct answer.

## 3 Probabilistic decision trees

**Theorem 1.** *There is a bounded error probabilistic decision tree computing the function $f(x_1,x_2) = x_1 \vee x_2$ with probability $\frac{3}{5}$.*

*Proof.* We reject with probability $\frac{1}{5}$. Simultaneously we query $x_1$ and $x_2$ with probability $\frac{2}{5}$. If the answer is 1, we accept. If the answer is 0, we accept and

reject with probability $\frac{1}{2}$.

| $x_1$ $x_2$ | 1 | 0 |
|---|---|---|
| 0 0 | 2/5 | 3/5 |
| 0 1 | 3/5 | 3/5 |
| 1 0 | 3/5 | 2/5 |
| 1 1 | 4/5 | 1/5 |

□

**Theorem 2.** *There is a bounded error probabilistic decision tree computing the function* $f(x_1, x_2, \ldots x_n) = x_1 \vee x_2 \vee \ldots \vee x_n$ *with probability* $\frac{n+1}{2n+1}$.

*Proof.* We reject with probability $\frac{1}{2n+1}$. Simultaneously we query $x_1$,$x_2$,...$x_n$ with probability $\frac{2}{2n+1}$. If the answer is 1, we accept. If the answer is 0, we accept and reject with probability $\frac{1}{2}$. The decision tree is symmetric, hence the result does not depend on order of variables.

| $x_1, x_2 \ldots x_n$ | 1 | 0 |
|---|---|---|
| 000..00 | n/2n+1 | n+1/2n+1 |
| 100..00 | n+1/2n+1 | n/2n+1 |
| 110..00 | n+2/2n+1 | n-1/2n+1 |
| ... | ... | ... |
| 111..10 | 2n-1/2n+1 | 2/2n+1 |
| 111..11 | 2n/2n+1 | 1/2n+1 |

The worst cases are the first and the second lines in table. Cut-point size is $\frac{1}{2n+1}$. □

**Theorem 3.** *There is a bounded error probabilistic decision tree computing the function* $f(x_1, x_2, \ldots x_n) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$ *with probability* $\frac{n+1}{2n+1}$.

*Proof.* We accept with probability $\frac{1}{2n+1}$. Simultaneously we query $x_1$,$x_2$,...$x_n$ with probability $\frac{2}{2n+1}$. If the answer is 0, we reject. If the answer is 1, we accept and reject with probability $\frac{1}{2}$. The decision tree is symmetric, hence the result does not depend on the order of variables.

| $x_1, x_2 \ldots x_n$ | 1 | 0 |
|---|---|---|
| 000..00 | 1/2n+1 | 2n/2n+1 |
| 100..00 | 2/2n+1 | 2n-1/2n+1 |
| ... | ... | ... |
| 111..10 | n/2n+1 | n+1/2n+1 |
| 111..11 | n+1/2n+1 | n/2n+1 |

The worst cases are the last and the second last lines in table. Cut-point size is $\frac{1}{2n+1}$. □

## 4 Quantum decision trees

For the functions OR, AND, PARTY, and MAJORITY, Beals *et al* [2] obtain the bounds in the table below.

| | exact | zero-error | bounded-error |
|---|---|---|---|
| OR,AND | $N$ | $N$ | $\Theta(\sqrt{(N)})$ |
| PARITY | $N/2$ | $N/2$ | $N/2$ |
| MAJORITY | $\Theta(N)$ | $\Theta(N)$ | $\Theta(N)$ |

**Theorem 4.** *There is a quantum decision tree with one query computing the function* $f(x_1, x_2) = x_1 \vee x_2$ *with probability* $\frac{5}{6}$.

*Proof.* First, we go to the states $Q_1$ with probability $\frac{1}{\sqrt{3}}$ and $Q_2$ with probability $\frac{\sqrt{2}}{\sqrt{3}}$. In state $Q_1$ we query each $x_i$ with amplitude $\frac{1}{\sqrt{2}}$. If the answer is 1, the query algorithm does to the state $q_{1i}$ with conditional amplitude +1. If the answer is 0, the query algorithm goes to the state $q_{1i}$ with conditional amplitude -1. In state $Q_2$ we query each $x_i$ with amplitude $\frac{1}{\sqrt{2}}$. If the answer is 1, the query algorithm does to the state $q_{2i_1}$. If the answer is 1, the query algorithm does to the state $q_{2i_0}$. Hence the distribution of amplitudes between $q_{11}$,$q_{12}$,$q_{21_1}$,$q_{21_0}$, $q_{22_1}$ and $q_{22_0}$ is as follows

| $x_1$ $x_2$ | $q_{11}$ | $q_{12}$ | $q_{21_1}$ | $q_{21_0}$ | $q_{22_1}$ | $q_{22_0}$ |
|---|---|---|---|---|---|---|
| 0 0 | $-\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | 0 | $+\frac{1}{\sqrt{3}}$ | 0 | $+\frac{1}{\sqrt{3}}$ |
| 0 1 | $-\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | 0 | $+\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{3}}$ | 0 |
| 1 0 | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{3}}$ | 0 | 0 | $+\frac{1}{\sqrt{3}}$ |
| 1 1 | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{3}}$ | 0 | $+\frac{1}{\sqrt{3}}$ | 0 |

After that we apply the Hadamard transform

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

to states $q_{11}$ and $q_{12}$, and the unitary transformation

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

to states $q_{21_1}$,$q_{21_0}$, $q_{22_1}$ and $q_{22_0}$. We get the amplitudes

| $x_1$ $x_2$ | $qr_1$ | $qr_2$ | $qr_3$ | $qr_4$ | $qr_5$ | $qr_6$ |
|---|---|---|---|---|---|---|
| 0 0 | $-\frac{1}{\sqrt{3}}$ | 0 | 0 | $+\frac{\sqrt{2}}{\sqrt{3}}$ | 0 | 0 |
| 0 1 | 0 | $-\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ |
| 1 0 | 0 | $+\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ |
| 1 1 | $+\frac{1}{\sqrt{3}}$ | 0 | $+\frac{\sqrt{2}}{\sqrt{3}}$ | 0 | 0 | 0 |

(**).

We accept in states $qr_2$,$qr_3$,$qr_5$ and $qr_6$ and reject in state $qr_4$, but in state $qr_1$ we go to the states $qr_{10}$ and $qr_{11}$ with probability $\frac{1}{\sqrt{2}}$. The state $qr_{11}$ is an accepting state and the state $qr_{10}$ is a rejecting state.

| $x_1\ x_2$ | $qr_{11}[1]$ | $qr_{10}[0]$ | $qr_2[1]$ | $qr_3[1]$ | $qr_4[0]$ | $qr_5[1]$ | $qr_6[1]$ |
|---|---|---|---|---|---|---|---|
| 0 0 | $-\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | 0 | 0 | $+\frac{\sqrt{2}}{\sqrt{3}}$ | 0 | 0 |
| 0 1 | 0 | 0 | $-\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ |
| 1 0 | 0 | 0 | $+\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ |
| 1 1 | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | 0 | $+\frac{\sqrt{2}}{\sqrt{3}}$ | 0 | 0 | 0 |

Measuring all these states, we get the result:

| $x_1\ x_2$ | 1 | 0 |
|---|---|---|
| 0 0 | 1/6 | 5/6 |
| 0 1 | 5/6 | 1/6 |
| 1 0 | 5/6 | 1/6 |
| 1 1 | 5/6 | 1/6 |

□

**Theorem 5.** *There is a quantum decision tree with one query computing the function $f(x_1, x_2) = x_1 \wedge x_2$ with probability $\frac{5}{6}$.*

*Proof.* The decision tree is similar to the last decision tree (see until the mark (**)). We accept in states $qr_3$ and reject states $qr_4$, $qr_5$ and $qr_6$, but in state $qr_1$ we go to the states $qr_{10}$ and $qr_{11}$ with probability $\frac{1}{\sqrt{2}}$. The state $qr_{11}$ is an accepting state and the state $qr_10$ is a rejecting state.

| $x_1\ x_2$ | $qr_{11}[1]$ | $qr_{10}[0]$ | $qr_2[0]$ | $qr_3[1]$ | $qr_4[0]$ | $qr_5[0]$ | $qr_6[0]$ |
|---|---|---|---|---|---|---|---|
| 0 0 | $-\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | 0 | 0 | $+\frac{\sqrt{2}}{\sqrt{3}}$ | 0 | 0 |
| 0 1 | 0 | 0 | $-\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ |
| 1 0 | 0 | 0 | $+\frac{1}{\sqrt{3}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | $-\frac{1}{\sqrt{6}}$ |
| 1 1 | $+\frac{1}{\sqrt{6}}$ | $+\frac{1}{\sqrt{6}}$ | 0 | $+\frac{\sqrt{2}}{\sqrt{3}}$ | 0 | 0 | 0 |

Measuring all these states, we get the result:

| $x_1\ x_2$ | 1 | 0 |
|---|---|---|
| 0 0 | 1/6 | 5/6 |
| 0 1 | 1/6 | 5/6 |
| 1 0 | 1/6 | 5/6 |
| 1 1 | 5/6 | 1/6 |

□

**Theorem 6.** [5] *There is a quantum decision tree with one query computing the function $j(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge x_3) \vee (\overline{x}_1 \wedge \overline{x}_2 \wedge \overline{x}_3)$ with probability $\frac{8}{9}$.*

*Proof.* We query $x_k$, $k = 1, 2, 3$, with amplitudes $\frac{1}{\sqrt{3}}e^0$, $\frac{1}{\sqrt{3}}e^{i\frac{2\pi}{3}}$, $\frac{1}{\sqrt{3}}e^{i\frac{4\pi}{3}}$, respectively. If the answer is 0, we go to the state $q_k$ with conditional amplitude -1. If the answer is 1, we go to the state $q_k$ with amplitude +1. Next, we perform the Fourier transform

$$\begin{pmatrix} \frac{1}{\sqrt{3}}e^0 & \frac{1}{\sqrt{3}}e^0 & \frac{1}{\sqrt{3}}e^0 \\ \frac{1}{\sqrt{3}}e^0 & \frac{1}{\sqrt{3}}e^{i\frac{2\pi}{3}} & \frac{1}{\sqrt{3}}e^{i\frac{4\pi}{3}} \\ \frac{1}{\sqrt{3}}e^0 & \frac{1}{\sqrt{3}}e^{i\frac{4\pi}{3}} & \frac{1}{\sqrt{3}}e^{i\frac{2\pi}{3}} \end{pmatrix}$$

On the states $q_1, q_2, q_3$, the first two outcome states are rejecting, the last is accepting.

| $x_1\ x_2\ x_3$ | $q_1[0]$ | $q_2[0]$ | $q_3[1]$ |
|---|---|---|---|
| 0 0 0 | $-\frac{1}{3}e^0 - \frac{1}{3}e^{i\frac{2\pi}{3}} - \frac{1}{3}e^{i\frac{4\pi}{3}}$ | $-\frac{1}{3}e^0 - \frac{1}{3}e^{i\frac{4\pi}{3}} - \frac{1}{3}e^{i\frac{2\pi}{3}}$ | $-\frac{1}{3}e^0 - \frac{1}{3}e^0 - \frac{1}{3}e^0$ |
| 0 0 1 | $-\frac{1}{3}e^0 - \frac{1}{3}e^{i\frac{2\pi}{3}} + \frac{1}{3}e^{i\frac{4\pi}{3}}$ | $-\frac{1}{3}e^0 - \frac{1}{3}e^{i\frac{4\pi}{3}} + \frac{1}{3}e^{i\frac{2\pi}{3}}$ | $-\frac{1}{3}e^0 - \frac{1}{3}e^0 + \frac{1}{3}e^0$ |
| 0 1 1 | $-\frac{1}{3}e^0 + \frac{1}{3}e^{i\frac{2\pi}{3}} + \frac{1}{3}e^{i\frac{4\pi}{3}}$ | $-\frac{1}{3}e^0 + \frac{1}{3}e^{i\frac{4\pi}{3}} + \frac{1}{3}e^{i\frac{2\pi}{3}}$ | $-\frac{1}{3}e^0 + \frac{1}{3}e^0 + \frac{1}{3}e^0$ |
| 1 1 1 | $\frac{1}{3}e^0 + \frac{1}{3}e^{i\frac{2\pi}{3}} + \frac{1}{3}e^{i\frac{4\pi}{3}}$ | $\frac{1}{3}e^0 + \frac{1}{3}e^{i\frac{4\pi}{3}} + \frac{1}{3}e^{i\frac{2\pi}{3}}$ | $\frac{1}{3}e^0 + \frac{1}{3}e^0 + \frac{1}{3}e^0$ |

Measuring all these states, we get the result:

| $x_1\ x_2\ x_3$ | $q_1[0]$ | $q_2[0]$ | $q_3[1]$ |
|---|---|---|---|
| 0 0 0 | 0 | 0 | 1 |
| 0 0 1 | 4/9 | 4/9 | 1/9 |
| 0 1 1 | 4/9 | 4/9 | 1/9 |
| 1 1 1 | 0 | 0 | 1 |

We accept in the cases $x_1 = x_2 = x_3 = 0$ and $x_1 = x_2 = x_3 = 1$ with probability 1. We reject in all the other cases with probability $\frac{8}{9}$. □

## References


1. A. Bērziņa, R. Bonner, and R. Freivalds. *Parameters in Ambainis-Freivalds algorithm.* Proc, Int. Workshop on Quantum Computation and Learning, 27-29 May 2000, Sweden, 2000, pp. 101 - 109.
2. R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. *Quantum lower bounds by polynomials.* Proc. 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS '98), pp. 352-361.
3. H. Buhrman, R. Cleve, R. de Wolf, C. Zalka. *Bounds for Small-Error and Zero-Error Quantum Algorithms.* Proc. 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '99), pp. 358-368.
4. D. Deutsch and R. Jozsa. *Rapid Solution of Problems by Quantum Computation.* Proc. Roy. Soc. Lond. A, 439, 1992. pp. 553-558.
5. R. Freivalds and L. Lāce. *Better upper bounds for advantages of probabilistic and quantum decision trees.* Unpublished manuscript, 2002.
6. R. Freivalds and A. Winter. *Quantum finite state transducers.* Lecture Notes in Computer Science 2234, Springer, 2001, pp. 233-242.
7. L. K. Grover. *A fast quantum mechanical algorithm for database search.* Proc. 28th Annual ACM Symposium on Theory of Computing (STOC), 1996, pp. 212-219.

8. J. Gruska. *Quantum Computing.* McGraw Hill, 1999.
9. P. Hajnal. *Decision tree complexity of Boolean functions.* Coll. Math. Soc. Janos Bolyai 60, Sets, graphs and numbers, Budapest, 1991; (1992), pp. 365-389.
10. M. Hirvensalo. *Quantum Computing.* Springer, 2001.
11. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.
12. N. K. Vereshchagin. *Randomized boolean decision trees: Several remarks.* Theoretical Computer Science 207, 1998, pp. 329-342.

# A Quantum Single-Query Automaton

Dmitry Kuzmenko*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
morfizm@yahoo.com

**Abstract.** We construct a quantum query automaton, which computet the Boolean function

$$f_4(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4) \vee (x_4 \wedge x_1)$$

with a single query. We also indicate how to generalize this construction for more general Boolean functions in DNF form, with $k \geq 4$ disjunctions and one or two variables in each conjunction term.

## 1 Introduction

For the definition of quantum query automata, see [25]. Complexity measures, limitations and algorithms for quantum query automata are discussed in [4, 2, 10, 5]. Here, we build an automaton, which computes the Boolean function

$$f_4(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4) \vee (x_4 \wedge x_1) \qquad (1)$$

in a single query, with probability $\frac{5}{9}$.

Recall briefly the connection of decision trees (devices for computing Boolean functions) and finite automata (devices for language recognition). Specifying a Boolean function of $n$ binary variables is obviously the same as specifying the finite language in $\{0,1\}^n$ where the function takes on value 1, so Boolean function computation and finite language recognition are one and the same problem. For the function (1), for example, the corresponding language consists of all binary words of length four with at least two cyclically consecutive letters equal to one.

A deterministic decision tree computing $f$ is then a direct encoding of a corresponding automaton: the paths through the tree are the computational paths of the automaton taken on different inputs, each path ending in an accepting state 1 or rejecting state 0. The probabilistic case is also straightforward, as it simply introduces a probability distribution over each of the two classes of deterministic computing devices. In the quantum case, the decision tree for computing a Boolean function is *defined* in terms of a finite quantum (query) automaton [25] by specifying for every input word $x \in \{0,1\}^n$ a unitary operator

$$A_x = U_T O_x U_{T-1} \ldots O_x U_1 O_x U_0, \qquad (2)$$

* Research supported by contract IST–1999–11234 (QAIP) from the European Commision and grant no. 01.0354 from the Latvian Council of Science.

where all $U_t$, $0 \leq t \leq T$, and $O_x$ are unitary, acting on some $\mathbb{C}^N$, in which the superpositions of the letters in an input word are a tensor factor. The output of computation is the result of measurement of the final superposition $A_x z_0$, $z_0$ being a fixed initial superposition, with respect to two subspaces representing the binary outcomes.

For the automaton, which we we are about to construct, $T = 1$; the operators $U_0$, $O_x$, $U_1$, and the final measurement operators will be described in terms of their action on tensor factors of the superposition space.

## 2 Quantum query automaton for $f_4$

### 2.1 Structure

The structure of the automaton is depicted in Figure 1 on page 46. There, the circles represent queries $x_k$, where the amplitude is moved to outgoing states, depending on the outcome of the query, $1 \leq k \leq 4$. The squares in the first column symbolize the application of $2 \times 2$ unitary matrix $Q$, followed by a measurement of the result. The rectangles in the third column stand for the application of a $3 \times 3$ matrix $M$. When a new state is introduced inside the tree, it has a starting amplitude of 0 (for example, in $q_3, q_6, q_{15}$).

The matrix $Q$ is of form

$$Q = Q_b = \begin{pmatrix} \sqrt{b} & \sqrt{1-b} \\ \sqrt{1-b} & -\sqrt{b} \end{pmatrix}; \tag{3}$$

for the pre-calculated value $b = \frac{4}{9}$,

$$Q = Q_{\frac{4}{9}} = \begin{pmatrix} \frac{2}{3} & \frac{\sqrt{5}}{3} \\ \frac{\sqrt{5}}{3} & -\frac{2}{3} \end{pmatrix}. \tag{4}$$

The matrix $M$ is a product of two rotations in 3-space, $M = AB$, where $A$ rotates in the first two co-ordinates by $\frac{\pi}{4}$, and $B$ rotates in the first and the third co-ordinates by an angle $\alpha$ such that $\cos\alpha = \frac{\sqrt{7}}{3}$ and $\sin\alpha = \frac{\sqrt{2}}{3}$. It is easily verified that

$$M = \begin{pmatrix} \frac{\sqrt{14}}{6} & -\frac{\sqrt{2}}{2} & -\frac{1}{3} \\ \frac{\sqrt{14}}{6} & \frac{\sqrt{2}}{2} & -\frac{1}{3} \\ \frac{\sqrt{2}}{3} & 0 & \frac{\sqrt{7}}{3} \end{pmatrix}. \tag{5}$$

### 2.2 Recognition probabilities

The matrix $Q$ is applied only when a query $x$ returns 0; it will act only on the vector $(1, 0)$. So, in this case, the result 1 has probability $b$, and the result 0 has probability $1 - b$.

The following list contains all possible inputs and outputs for the matrix $M$. When probabilities of outcome 1 and 0, respectively, are calculated, we take into

account all subtrees which begin at the matrix $M$. For the states $q_{16}$, $q_{17}$, $q_{18}$, and $q_{25}$, for example, putting $v = (q_{16}, q_{17}, q_{18})$ and $v' = vM$, the probability of outcome 1 is $|v'_1|^2 + \frac{1}{2}|v'_3|^2$.

| case | $x_1x_2$ | $v = (q_{16}, q_{17}, q_{18}$ | $vM$ | prob. of 1 | prob. of 0 |
|---|---|---|---|---|---|
| ♯1 | 00 | $(0, 0, 0)$ | $(0, 0, 0)$ | 0 | 0 |
| ♯2 | 01 | $(0, \frac{1}{2\sqrt{2}}, 0)$ | $(\frac{\sqrt{7}}{12}, \frac{1}{4}, -\frac{1}{6\sqrt{2}})$ | $\frac{1}{18}$ | $\frac{5}{72}$ |
| ♯3 | 10 | $(\frac{1}{2\sqrt{2}}, 0, 0)$ | $(\frac{\sqrt{7}}{12}, -\frac{1}{4}, -\frac{1}{6\sqrt{2}})$ | $\frac{1}{18}$ | $\frac{5}{72}$ |
| ♯4 | 11 | $(\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, 0)$ | $(\frac{\sqrt{7}}{6}, 0, -\frac{1}{3\sqrt{2}})$ | $\frac{2}{9}$ | $\frac{1}{36}$ |

Matrix Q has only one input and output case:

| case | $x_1$ | $v = (q_2, q_3)$ | $vQ$ | prob. of 1 | prob. of 0 |
|---|---|---|---|---|---|
| ♯5 | 0 | $(\frac{1}{2}, 0)$ | $(\frac{1}{3}, \frac{\sqrt{5}}{6})$ | $\frac{1}{9}$ | $\frac{5}{36}$ |

Finally, we list all input combinations ($x$ values), and the corresponding final result probabilities (cyclically shifted combinations are considered identical, e.g. $0110 = 0011$).

| $x$ | cases in use | prob. of 1 | prob. of 0 |
|---|---|---|---|
| 0000 | ♯5 ♯5 ♯5 ♯5 | $\frac{4}{9}\ (\frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9})$ | $\frac{5}{9}\ (\frac{5}{36} + \frac{5}{36} + \frac{5}{36} + \frac{5}{36})$ |
| 1000 | ♯2 ♯3 ♯5 ♯5 ♯5 | $\frac{4}{9}\ (\frac{1}{18} + \frac{1}{18} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9})$ | $\frac{5}{9}\ (\frac{5}{72} + \frac{5}{72} + \frac{5}{36} + \frac{5}{36} + \frac{5}{36})$ |
| 1100 | ♯4 ♯2 ♯3 ♯5 ♯5 | $\frac{5}{9}\ (\frac{2}{9} + \frac{1}{18} + \frac{1}{18} + \frac{1}{9} + \frac{1}{9})$ | $\frac{4}{9}\ (\frac{1}{36} + \frac{5}{72} + \frac{5}{72} + \frac{5}{36} + \frac{5}{36})$ |
| 1010 | ♯2 ♯3 ♯2 ♯3 ♯5 ♯5 | $\frac{4}{9}\ (4\frac{1}{18} + \frac{1}{9} + \frac{1}{9})$ | $\frac{5}{9}\ (4\frac{5}{72} + \frac{5}{36} + \frac{5}{36})$ |
| 1110 | ♯2 ♯3 ♯4 ♯4 ♯5 | $\frac{6}{9}\ (\frac{1}{18} + \frac{1}{18} + \frac{2}{9} + \frac{2}{9} + \frac{1}{9})$ | $\frac{3}{9}\ (\frac{5}{72} + \frac{5}{72} + \frac{1}{36} + \frac{1}{36} + \frac{5}{36})$ |
| 1111 | ♯4 ♯4 ♯4 ♯4 | $\frac{8}{9}\ (\frac{2}{9} + \frac{2}{9} + \frac{2}{9} + \frac{2}{9})$ | $\frac{1}{9}\ (\frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36})$ |

Comparing the probabilities we see that for all words with at least two letters 1 in succession, the probability of outcome 1 is at least $\frac{5}{9}$; otherwise, the probability of 0 is also also at least $\frac{5}{9}$.

## 3 Discussion

### 3.1 Finding $Q$ and $M$

How were the matrixes $Q$ and $M$ found? Have a look at the first table: there are two input states[1] for matrix $M$, and each of them can have amplitude $\frac{1}{2\sqrt{2}}$ or 0. We need a transformation that gives a higher probability of outcome 1 than 0, $p(1) > p(0)$, when the input vector is $(\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}})$ (let us call it the 'good case'), but $p(1) < p(0)$ for all other cases: $(\frac{1}{2\sqrt{2}}, 0)$ and $(0, \frac{1}{2\sqrt{2}})$ (call this the 'bad case'). Denote $p(1)$ by $g$ in the good case, $g > \frac{1}{2}$, and by $b$ in the bad case, $b < \frac{1}{2}$.

If there are only bad cases, then the total probability of the result 0 is $1 - b$. But, if there is at least one good case, the total probability of 1 should be greater

[1] At this moment, the third additional state, e.g. $q_{15}$, is not taken into account.

than half, in spite of all other cases being bad. So, $g$ should be big enough, to compensate $b$ in other places.

Let the radius of isolation of the automaton be $\epsilon$ (then the error is $\frac{1}{2} - \epsilon$). The worst case when result 1 must be given is a state $(\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}})$ at one matrix $M$ (hence, the sum of the squares of amplitude moduli is $\frac{1}{8} + \frac{1}{8} = \frac{1}{4}$), and bad cases in all other places (sum of squares of moduli is $1 - \frac{1}{4} = \frac{3}{4}$). This gives the inequality

$$\frac{1}{4}\, g + \frac{3}{4}\, b \geq \frac{1}{2} + \epsilon. \tag{6}$$

Note that this is the worst case; if there is one 'good case', say the input is 0111, we have a weaker inequality: $\frac{1}{2}\, g + \frac{1}{2}\, b \geq \frac{1}{2} + \epsilon$, and, for example, 1111 will give a still weaker inequality $1\, g \geq \frac{1}{2} + \epsilon$.

Similarly, the constraint for all bad cases (result 0) is

$$b \leq \frac{1}{2} - \epsilon. \tag{7}$$

It only remains to find the largest possible $\epsilon$, and out of that, the values of $g$ and $b$. Necessary probability ratio can be found with unitary rotations. All input cases for the matrix $M$ are listed in the first table, above; we may omit the case ♯1, $(0, 0)$, as the allocation of the probabilities of the results 1 and 0 occurs then in other places ($M$ does not change anything in case ♯1). Write vectors in the form $(x, y)$; after a transformation, we will measure the result 1 at $x$, and the result 0 at $y$.

We need the $x$ co-ordinate of a case ♯4 vector to be greater than in all other cases. To achieve this, rotate a vector by $\frac{\pi}{4}$; now the ratio $p(1) : p(0)$ in case ♯4 is $1 : 0$, but in all other cases it is $\frac{1}{2} : \frac{1}{2}$ ($g = 1$, $b = \frac{1}{2}$).

We need $b < \frac{1}{2}$. Add a new state $z$ and make a rotation in the $x, z$ plane (there is no difference in which direction we rotate). As a side effect, there appears an amplitude $\sin^2 \alpha$ in the $z$-variable. We can distribute it equally between results 1 and 0.

Now the ratio $p(1) : p(0)$ is $(\cos^2 \alpha + \frac{1}{2} \sin^2 \alpha) : \frac{1}{2} \sin^2 \alpha$ in case ♯4, while in cases ♯2 and ♯3, it is $(\frac{1}{2} \cos^2 \alpha + \frac{1}{4} \sin^2 \alpha) : (\frac{1}{2} + \frac{1}{4} \sin^2 \alpha)$. Case ♯4 is 'good', hence

$$g = \cos^2 \alpha + \frac{1}{2} \sin^2 \alpha = \frac{1}{2} + \frac{1}{2} \cos^2 \alpha, \tag{8}$$

but cases ♯2 and ♯3 are 'bad', hence

$$b = \frac{1}{2} \cos^2 \alpha + \frac{1}{4} \sin^2 \alpha = \frac{1}{4} + \frac{1}{4} \cos^2 \alpha. \tag{9}$$

Inserting these into (6) and (7), respectively, we can bound $\alpha$

$$\frac{3}{5} + \frac{16}{5}\epsilon \leq \cos^2 \alpha \leq 1 - 4\epsilon. \tag{10}$$

This gives an estimate on $\epsilon$:

$$\epsilon \leq \frac{1}{18}. \tag{11}$$

For the maximum radius of isolation $\epsilon = \frac{1}{18}$ (giving the correct result with probability $\frac{5}{9}$), the estimate (10) gives, up to a choice of sign, $\cos\alpha = \frac{\sqrt{7}}{2}$ and $\sin\alpha = \frac{\sqrt{2}}{3}$, and hence, by (6) and (7), $g = \frac{8}{9}$ and $b = \frac{4}{9}$

The cases the matrix $Q$ operates on are always 'bad', so $Q$ should distribute probabilities $p(1)$ and $p(0)$ with ratio $p(1) : p(0)$ equal to $b : 1 - b$, that is, $(1,0)Q = (\sqrt{b}, \sqrt{(1-b)})$; an appropriate matrix Q is thus of form (3), which, for $b = \frac{4}{9}$ is the matrix (4).

### 3.2 Generalizations

This algorithm can be generalized. For any Boolean function, which can be represented in disjunctive normal form (DNF) with one or two arguments in each conjunction, we can build a single-query quantum automaton. As for the case of $f_4$, there will be a matrix $M$ for each conjunction term of the DNF. Let $k$ be the number of terms in the DNF, which is also the number of matrices $M$. Then inequality (6) now takes the form

$$g\frac{1}{k} + b(1 - \frac{1}{k}) \geq \frac{1}{2} + \epsilon. \tag{12}$$

Inserting (8) and (9), together with (7), this gives

$$\frac{k-1}{k+1} + \frac{4k}{k+1}\,\epsilon \leq \cos^2\alpha \leq 1 - 4\,\epsilon, \tag{13}$$

recovering (10) if $k = 4$. This gives, as previously (11), an estimate on $\epsilon$:

$$\epsilon \leq \frac{1}{4k+2}. \tag{14}$$

Taking, as previously, the largest value $\epsilon = \frac{1}{4k+2}$ in (13) gives $\cos^2\alpha = \frac{2k-1}{2k+1}$ and $\sin^2\alpha = \frac{2}{2k+1}$, and hence, by (8) and (9), $g = \frac{2k}{2k+1}$ and $b = \frac{1}{2k+1}$.

## References

1. A. Ambainis. *Quantum lower bounds by quantum arguments.* 2000. http://citeseer.nj.nec.com/432640.html, http://www.cs.berkeley.edu/ ambainis/ps/density4.ps.gz
2. A. Ambainis. *Quantum Query Model: Algorithms and Lower Bounds.* 1999. http://www.ima.mdh.se/personal/rbr/courses/riga99proc.html, http://www.ima.mdh.se/personal/rbr/courses/riga99proc/ambainis.ps
3. R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. *Quantum lower bounds by polynomials.* Proc. FOCS'98, pp. 351-361. http://citeseer.nj.nec.com/51642.html, http://www.cwi.nl/ rdewolf/publ/qc/focs98.ps.gz
4. A. Berzina, R. Bonner and R. Freivalds. *Parameters in Ambainis-Freivalds algorithm.* In: R. Bonner and R. Freivalds, Quantum Computation and Learning, Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 101 - 109.

5. H. Buhrman, R. Cleve, R. de Wolf and C. Zalka. *Bounds for small-error and zero-error quantum algorithms.* 1998. http://xxx.lanl.gov/abs/quant-ph/9802062, http://citeseer.nj.nec.com/198249.html, http://www.cwi.nl/ rdewolf/publ/qc/focs99.ps.gz
6. H. Buhrman and R. de Wolf. *Complexity Measures and Decision Tree Complexity: A Survey.* 2000. http://citeseer.nj.nec.com/buhrman00complexity.html, http://www.cwi.nl/ rdewolf/publ/qc/dectree.ps.gz
7. R. Freivalds and L. Lace. *Better upper bounds for advantages of probabilistic and quantum decision trees.* Unpublished manuscript, 2002.

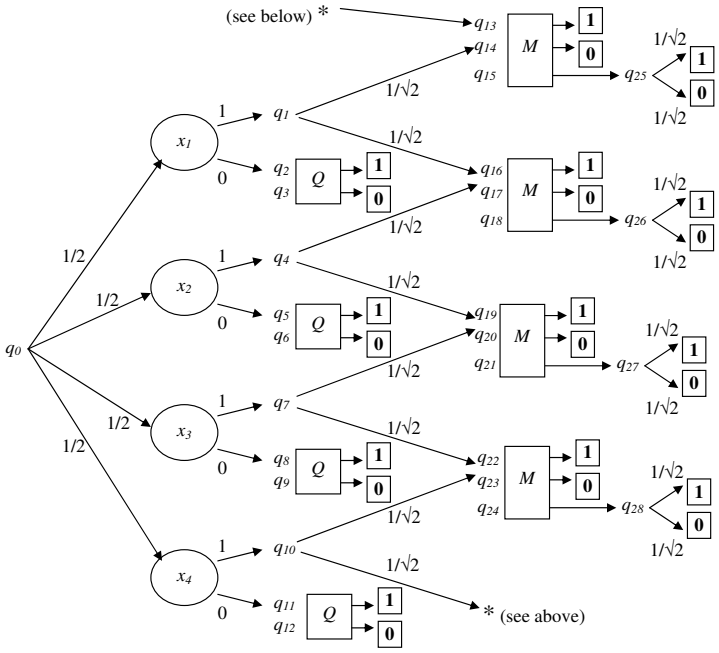


**Fig. 1.** The structure of the single-query automaton

# Deterministic Finite Automata and 1-way Measure-Many Quantum Finite Automata with Non-isolated Cutpoint

Raitis Ozols*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
sm01022@lanet.lv

**Abstract.** We exhibit a class of languages not recognized by deterministic finite automata (DFA), but recognized by 1-way measure-many quantum finite automata (QFA) with non-isolated cut-point.

## 1 A class of languages not recognized by DFA

Let $\mathbb{N}$ be the set of all non-negative integers, and let $\pi\mathbb{Q}$ be the set of all rational multiples of $\pi$.

**Lemma 1.** *For any real $\alpha, \beta$ such that $\alpha \notin \pi\mathbb{Q}$, the sequence $\cos(\alpha n+\beta)$, $n \in \mathbb{N}$, is dense in the interval $[-1, 1]$.*

*Proof.* It is known and straightforward that for $\alpha \notin \pi\mathbb{Q}$ the sequence $e^{i\alpha n}$, $n \in \mathbb{N}$, is dense (indeed, uniformly distributed) in the complex unit circle; the conclusion follows by rotating the circle by $e^{ib}$ and projecting it onto the real axis.

**Theorem 1.** *Let $\alpha, \beta$ be a real numbers, $\alpha \notin \pi\mathbb{Q}$, let $K$ be a subset of the interval $[-1, 1]$. Define a language over the alphabet $\{a\}$ by*

$$L = L_{\alpha,\beta,K} = \{a^n : n \in \mathbb{N},\ \cos(n\alpha + \beta) \in K\}.$$

*If $K$ is not dense in $[-1, 1]$, then $L$ is not recognized by any deterministic finite automaton.*

*Proof.* Clearly, the sequence of states taken on by a deterministic finite automaton (DFA) $A$ upon reading a long enough word $a^n$ will eventually become periodic; let the length of the period be $N$. Hence, for $n$ large enough, if $A$ accepts a word $a^n$ it will also accept all words $a^{n+kN}$, $k \in \mathbb{N}$. But, by Lemma 3, the sequence $\cos((n + kN)\alpha) = \cos(k(N\alpha) + n\alpha)$, $k \in \mathbb{N}$, is dense in $[-1, 1]$, and since $K$ is not dense in $[-1, 1]$, some of the words $a^{n+kN}$ will not be in $L$. □

*Remark.* If $K$ contains the interval $(-1, 1)$, the condition $\cos(n\alpha + \beta) \in K$ is empty, and hence $L_{\alpha,\beta,K} = \{a^n : n \in \mathbb{N}\}$ is trivially recognized by DFA.

* Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission.

## 2 Recognition by QFA with non-isolated cut-point

An (probabilistic) automaton is said to recognize a language $L$ with non-isolated cut-point $p$ ($p \in [0,1]$) if it accepts every $\omega \in L$ with probability $p(\omega) > p$, and it accepts every $\omega \notin L$ with probability $p(\omega) < p$.

For real $a$, denote by $\rho_a$ the two-by-two symmetric matrix with both terms on the diagonal equal to $\cos a$, and the remaining terms equal to $i \sin a$, $i = \sqrt{-1}$. It is easy to verify that $\rho_a \rho_b = \rho_{a+b}$, and, in particular, $\rho_a^n = \rho_{an}$, $n \in \mathbb{N}$.

**Proposition 1.** *For any real $\alpha, \beta$ such that $\alpha \notin \pi\mathbb{Q}$, and any non-negative $p_1$, $p_2$, $p_3$, such that $p_1 + p_2 + p_3 \leq 1$, there exists a 1-way measure-many quantum finite automaton, which accepts words $a^n$ over a one-letter alphabet $\{a\}$ with probability $p(a^n) = p_3 + p_2 + (p_1 - p_2)\cos^2(n\alpha + \beta)$, $n \in \mathbb{N}$.*

*Proof.* Consider a QFA with four states: $q_1$ and $q_2$ - non-halting, $q_3$ - accepting, and $q_4$ - rejecting. Define the (unitary) transition matrices $U_\sharp$, $U_a$, and $U_\$$, corresponding to the left end-marker $\sharp$, the symbol $a$, and the right end-marker \$, respectively, as follows.

The first row of $U_\sharp$ is any real vector $z = (z_1, z_2, z_3, z_4)$ of norm one such that $z_j^2 = p_j$, $j = 1, 2, 3$; the remaining rows of $U_\sharp$ are arbitrary subject to the unitarity condition. The matrix $U_a = \rho_\alpha \otimes I_2$, ie $U_a$ has blocks $\rho_\alpha$ and the 2-dimensional identity matrix $I_2$ on the sinister diagonal, and the remaining entries are zero. Finally, the matrix $U_\$$ has similar blocks as $U_a$, but on the dexter diagonal: the right uppermost block is $\rho_\beta$, the left lowermost block is $I_2$, and the remaining entries are zero.

To describe the operation of the automaton, it will be convenient to write vectors $x = (x_1, x_2, x_3, x_4)$ in the form $x = (x', x'')$ with $x' = (x_1, x_2)$ and $x'' = (x_3, x_4)$.

Let the initial distribution of amplitudes be $v = (1, 0, 0, 0)$. After reading the left end-marker $\sharp$, the distribution of amplitudes becomes $vU_\sharp = z$. At this moment, the probabilities of an acceptance and rejection are equal to $p_3$ and $p_4$, respectively, and, up to normalization, new distribution of amplitudes is $(z_1, z_2, 0, 0) = (z', 0)$. Since vectors of this form are acted upon only by the first block $\rho_\alpha$ of $U_a$, the subsequent $n$-fold reading of the symbol $a$ will result in multiplying the vector $z' = (z_1, z_2)$ of the first two amplitude coordinates by the matrix $\rho_\alpha^n = \rho_{n\alpha}$, the two other coordinates remaining zero. Note, that all intermediate measurements are trivial, giving zero probabilities to acceptance/rejection, and not changing the amplitude distribution. Finally, the reading of the right end-marker \$ sends a distribution $(x', 0)$ to $(0, \rho_\beta x')$; consequently the final distribution after having read the word $\sharp a^n \$$ will be $(0, \rho_\beta \rho_{n\alpha} z') = (0, \rho_{n\alpha+\beta} z')$. Hence, the total probability of acceptance of the word $a^n$ will be

$$p_3 + |z_3 \cos(n\alpha + \beta) + i z_2 \sin(n\alpha + \beta)|^2 = p_3 + p_2 + (p_1 - p_2)\cos^2(n\alpha + \beta),$$

as claimed. □

Before stating the main theorem, we make a technical observation needed in its proof.

**Lemma 2.** *If $\alpha, \beta$ are real numbers such that $\alpha+\beta \notin \pi\mathbb{Q}$ and $\alpha-\beta \notin \pi\mathbb{Q}$, then $\cos^2\alpha \neq \cos^2\beta$.*

*Proof.* Write $\cos^2\alpha - \cos^2\beta$ as a product of the sine and cosine functions of the angles $\frac{\alpha+\beta}{2}$ and $\frac{\alpha-\beta}{2}$. □

**Theorem 2.** *For every $p \in (0,1)$ there exists a language, which can be recognized by some 1-way measure-many quantum finite automaton with non-isolated cut-point $p$, but cannot be recognized by any deterministic finite automaton.*

*Proof.* Let $A = A_{p_1,p_2,p_3,\alpha,\beta}$ be an automaton ascertained by Proposition 1. Clearly, a necessary condition for $A$ to accept a language over the one-letter alphabet $\{a\}$ with non-isolated cut-point $p$, is that the probability of acceptance $p(a^n)$ of words $a^n$ never equals $p$ :

$$p(a^n) = p_3 + p_2 + (p_1 - p_2)\cos^2(n\alpha + \beta) \neq p \tag{1}$$

for all $n \in \mathbb{N}$. Choose real $\alpha \notin \pi\mathbb{Q}$, $b \in \mathbb{Q}$, $\beta = \alpha + \pi b$, and $z_1 \neq z_2$. Condition (1) now reads:

$$\cos^2(n\alpha + \alpha + \pi b) \neq \frac{p - (p_3 + p_2)}{p_1 - p_2}. \tag{2}$$

By Lemma 2, condition (2) will hold if its right side can be written as $\cos^2\theta$ for some $\theta \in \pi\mathbb{Q}$. To this effect, ensure that

$$p_3 + p_2 < p < p_3 + p_1, \tag{3}$$

so the right side of (2) lies in the interval $(0,1)$; it is clear by a density argument that for suitable $p_k$ satisfying (3) a $\theta \in \pi\mathbb{Q}$ may be found.

Let $L = \{a^n : n \in \mathbb{N},\ p(a^n) > p\}$. Clearly, $A$ recognizes $L$ with non-isolated cut-point $p$. Since another way of writing $p(a^n) > p$ is $\cos^2(n\alpha+\beta) > \cos^2\theta$, and $\cos^2\theta > 0$, Theorem 3 applies, and hence the language $L$ is not recognized by any deterministic finite automaton. □

## 3 Open problems

We have exhibited a class of languages not recognizable by DFA, but recognizable by MM-QFA with non-isolated cut-point.

Given such a language, it is natural to look for simplest possible QFA of this type, which recognize it. In particular, we would like to know whether all the elements of the matrices of the QFA could be of form $(a+bi)(\cos\alpha + i\sin\beta)$, with $a, b$ real and $\alpha, \beta \notin \pi\mathbb{Q}$.

Another open problem is the existence of a language not recognized by *probabilistic* finite automata, but recognized with non-isolated cut-point by MM-QFA.

## References

[ACLRS 96] A. Andzans, J. Cakste, T. Larfelds, L. Ramana and M. Seile. *Mean Value Method.* Riga, Zvaigzne ABC, 1996. (in Latvian)

[ABFK 99] A. Ambainis, R. Bonner, R. Freivalds and A. Kikusts. *Probabilities to Accept Languages by Quantum Finite Automata.* COCOON 1999, Lecture Notes in Computer Science 1627, 1999, pp. 174-183.

[AF 98] A. Ambainis and R. Freivalds. *1-way quantum finite automata: strengths, weaknesses and generalizations.* 39th FOCS, 1998, pp. 332-341. http://xxx.lanl.gov/abs/quant-ph/9802062

[Gr 99] J. Gruska. *Quantum Computing.* McGraw Hill, 1999.

[BBF00] A. Berzina, R. Bonner and R. Freivalds. *Parameters in the Ambainis-Freivalds algorithm.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning, Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 101-109.

# The Complexity of Probabilistic versus Quantum Finite Automata

Gatis Midrijānis*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
mgatis@inbox.lv

**Abstract.** A language $L_n$ is presented, which is recognizable by a probabilistic finite automaton with $O(log^2 n)$ states with probability $1-\epsilon$, for all $\epsilon > 0$, and by a deterministic finite automaton with $O(n)$ states, but any quantum finite automaton recognizing $L_n$ needs $2^{\Omega(n/\log n)}$ states.

## 1 Introduction

Probabilistic finite automata (PFA) generalize deterministic finite automata (DFA), and many authors, for example [2, 5, 7, 4], researched the advantages of PFA over DFA. On the other hand, it is known [3, 2] that the size of reversible and quantum finite automata (RFA and QFA, respectively) recognizing some regular (i.e. recognizable by DFA) languages grows almost exponentially. And so, Ambainis *et al* [3] write:

> Another open problem involves the blow up in size while simulating a 1-way PFA by a 1-way QFA. The only known for doing this is by simulating the PFA by a 1-way QFA and then simulating the DFA by a QFA. Both simulating a PFA by a DFA [7, 5, 6], and simulating DFA by a QFA (this paper) can involve exponential or nearly exponential increase in size. This means that the straightforward simulation of a probabilistic automaton by a QFA (described above) could result in a doubly-exponential increase in size. However, we do not known of any examples where both transforming a PFA into a DFA and transforming a DFA into a QFA cause big increases of size. Better simulations of PFA by QFAs may well be possible.

Presently we address this problem.

## 2 Definitions and known results

We define 1-way QFA (further in the text simply QFA) as in [2, 3]. This model, first introduced in [1], is not the most general one, but it is easy to implement

* Research supported by Grant No.01.0354 from the Latvian Council of Science, and Contract IST-1999-11234 (QAIP) from the European Commission.

and to deal with. A QFA has a finite set of basis states $Q$, consisting of tree parts: accepting states ($Q_{acc}$), rejecting states ($Q_{rej}$) and non-halting states ($Q_{non}$). One of the states, $q_{ini}$, is distinguished as the starting state. Inputs to a QFA are words over a finite alphabet $\Sigma$. We shall also use the symbols $\emptyset$ and \$ that do not belong to $\Sigma$ to denote the left and the right end marker, respectively. The set $\Gamma = \Sigma \cup \{\emptyset, \$\}$ denotes the working alphabet of the QFA. To each symbol $\sigma \in \Gamma$, corresponds a unitary transformation $U_\sigma$ on the space $\mathbb{C}^Q$.

At any time, the state of a QFA is a superposition of basis states in $Q$. The computation starts in the superposition $|q_{ini}\rangle$. Then, the transformations corresponding to the left end marker $\emptyset$, the letters of the input word $x$, and the right end marker \$, are applied in succession to the state of the automaton, unless a transformation results in acceptance or rejection of the input. A transformation consists of two steps: First, $U_\sigma$ is applied to $|\psi\rangle$, the current state of the automaton, to obtain the new state $|\psi'\rangle$. Then, $|\psi'\rangle$ is measured with respect to the observable $E_{acc} \oplus E_{rej} \oplus E_{non}$, where $E_i$ is the linear span in $\mathbb{C}^Q$ of the set $\{|q\rangle : q \in Q_i\}$, $i \in \{acc, rej, non\}$. The probability of observing $E_i$ is equal to the squared norm of the projection of $|\psi'\rangle$ onto $E_i$. On measurement, the state of the automaton 'collapses' to the projection onto the space observed, i.e., it becomes equal to the projection, normalized to a unit superposition. If we observe $E_{acc}$ (or $E_{rej}$), the input is accepted (or rejected). Otherwise, the computation continues, and the next transformation, if any, is applied.

A QFA is said to *accept* (or *recognize*) a language $L$ with probability $p > \frac{1}{2}$ if it accepts every word in $L$ with probability at least $p$, and rejects every word not in $L$ with probability at least $p$.

A RFA is a QFA with only the elements 0 and 1 in its transition matrices. A PFA is defined essentially as a QFA but with stochastic, instead of unitary, transition matrices. A DFA is a PFA with only 0 and 1 in the matrices. The *size* of a finite automaton is defined as the number of its (basis) states. More exact definitions can be found, for example, in [2].

Ambainis and Freivalds [2] give a language $L_n^\times$ consisting of a single word $a^n$ in a single-letter alphabet, and prove:

**Theorem 1.** *Any DFA recognizing $L_n^\times$ has at least $n$ states, but, for any $\epsilon > 0$, there is a PFA with $O(\log^2 n)$ states recognizing $L_n^\times$ with probability $1 - \epsilon$.*

*Proof.* We briefly sketch the idea. The first part is evident. To prove the second part, Freivalds [5] used the following construction. $O(\frac{\log n}{\log \log n})$ different primes are employed and $O(\log n)$ states are used for every employed prime. At first, the automaton randomly chooses a prime $p$, and the remainder modulo $p$ of the length of an input word is found and compared with the standard. Additionally, once in every $p$ steps a transition to a rejecting state is made with a 'small' probability proportional to $\frac{p}{n}$. The number of used primes suffices to assert that, for every input of length less than $n$, most of primes $p$ give remainders different from the remainder of $n$ modulo $p$. The 'small' probability is chosen to have the rejection high enough for every input length $N$ such both $N \neq n$ and $\epsilon$-fraction of all the primes used have the same remainders modulo $p$ as $n$. □

Ambainis *et al* [3] introduce the following definition:

**Definition 1.** *A function* $f : \{0,1\}^m \times R \longmapsto \mathbb{C}^{2^n}$ *serially encodes* $m$ *classical bits into* $n$ *qubits with* $p$ *success, if for any* $i \in \{1,\ldots,n\}$ *and* $b_{[i+1,n]} = b_{i+1}\ldots b_n \in \{0,1\}^{n-i}$, *there is a measurement* $\Theta_{i,b_{[i+1,n]}}$ *that returns* 0 *or* 1, *and has the property that for all* $b \in \{0,1\}^m$ *the probability that* $\Theta_{i,b_{[i+1,n]}}|f(b,r)\rangle = b_i$ *is at least* $p$.

They prove:

**Theorem 2.** *Any quantum serial encoding of* $m$ *bits into* $n$ *qubits with constant success probability* $p > 1/2$ *implies* $n \geq \Omega(\frac{m}{\log m})$.

They also define an $r$-restricted 1-way QFA for a language $L$ as a 1-way QFA that recognizes $L$ with probability $p > 1/2$, and which halts with non-zero probability before seeing the right end marker only after it has read $r$ letters of the input. They prove:

**Theorem 3.** *If there is a 1-way QFA with* $S$ *states recognizing a language* $L$ *with probability* $p$, *then there is an* $r$*-restricted 1-way QFA with* $O(rS)$ *states that recognizes* $L$ *with probability* $p$.

## 3 Results

The proof of Theorem 4 below will use the following lemma:

**Lemma 1.** *The language* $L_1 = \{\omega \in \{0,1\}^* : \exists x, y \in \{0,1\}^* : \omega = x00y\}$ *is recognizable by DFA.*

*Proof.* Consider an automaton with five states $q_0$, $q_1$, $q_2$, $q_{acc}$ and $q_{rej}$, and with transitions defined by: $f(q_0,0) = q_1$, $f(q_0,1) = q_0$, $f(q_1,0) = q_2$, $f(q_1,1) = q_0$, $f(q_2,0) = f(q_2,1) = q_2$, $f(q_0,\$) = f(q_1,\$) = q_{rej}$, and $f(q_2,\$) = q_{acc}$. □

**Theorem 4.** *For all* $k \geq 1$, $n = 2k$, *define the languages*

$$L_n = \{\omega \in \{0,1\}^n : \exists x, y \in \{0,1\}^* : \omega = x00y\}.$$

*Then,*

0. *There is a RFA (hence also a QFA, a PFA and a DFA) recognizing* $L_n$.
1. *Any RFA that recognizes* $L_n$, *has at least* $2^{O(n)}$ *states.*
2. *Any QFA that recognizes* $L_n$ *with probability* $p > 1/2$, *has at least* $2^{\Omega(\frac{n}{\log n})}$ *states.*
3. *Any DFA that recognizes* $L_n$, *has at least* $O(n)$ *states.*
4. *For any* $\epsilon > 0$, *there is a PFA with* $O(\log^2 n)$ *states recognizing* $L_n$ *with probability* $1 - \epsilon$.

*Proof.* 0. Every finite language is recognizable by some RFA, and $L_n$ is a finite language.

1. Let $M$ be a RFA recognizing $L_n$. We give $M$ the word $a_1 1 a_2 1 a_3 1 \dots a_k 1$, $a_i \in \{0,1\}$. It is obvious that $M$ cannot decide what to answer until the end of the word. It is also clear that $M$ must branch at every $a_i$. Indeed, if for some $i$ the automaton did move to the same state after reading $a_i = 0$ as after reading $a_i = 1$, then we could feed it $01^{n-2i}$ as the remaining symbols in the two words, only one of which is in $L_n$ but both lead to the same final state. When $M$ branches at $a_i$, we say that it 'remembers' this bit. But could it merge ('forget') afterwards? No, because merging states upon reading an input symbol is forbidden by reversibility, and merging states upon reading different input symbols is excluded for the same reason that causes branching. It then follows that $M$ remembers all the $k$ bits $a_i$, and the total number of its states is thus at least $2^k$.

2. We use the technique introduced by [3]. Let $M$ be any $n$-restricted QFA accepting $L_n$ with probability $p > \frac{1}{2}$. The following claim formalizes the intuition that the state of $M$ after $n$ symbols of form $a_1 1 a_2 1 a_3 1 \dots a_k 1$ have been read is an 'encoding' of the $\{a_i\}$ (for RFA, deterministic, we used the term 'remember').

*Claim.* There is a serial encoding of $k$ bits into $\mathbb{C}^Q$, and hence into $\lceil \log |Q| \rceil$ qubits, where $Q$ is the set of basis states of $M$.

*Proof of Claim.* Let $Q_{acc}$ and $Q_{rej}$ be the sets of accepting and rejecting states, respectively. Let $U_\sigma$ be the unitary operator of $M$ corresponding to the symbol $\sigma \in \{0, 1, \emptyset, \$\}$. We define an encoding $f : \{0,1\}^k \longrightarrow \mathbb{C}^Q$ of $k$-bit strings into unit superpositions over the basis states of $M$ by letting $|f(x)\rangle$ be the state of $M$ after the input string $a_1 1 a_2 1 a_3 1 \dots a_k 1$, $a_i \in \{0,1\}$, has been read. We assert that $f$ is a serial encoding. To show this, we exhibit a suitable measurement for the $a_i$th bit for every $i \in \{1, \dots k\}$. Let, for $y \in \{0,1\}^{n-2i+1}$, $V_i(y) = U_\$ U_1^{n-2i} U_0 U_{y_1}^{-1} U_{y_2}^{-1} \dots U_{y_{n-2i-1}}^{-1} U_{y_{n-2i}}^{-1} U_{y_{n-2i+1}}^{-1}$. The $i$th measurement then consists of first applying the unitary transformation $V_i(1 a_{i+1} 1 \dots 1 a_k 1)$ to $|f(x)\rangle$, and then measuring the resulting superposition with respect to $E_{acc} \otimes E_{rej} \otimes E_{non}$. Since for words of form $a_1 1 a_2 1 \dots 1 a_i 0 1^{n-2i}$, membership in $L_n$ is decided by the $a_i$, and because such words are accepted or rejected by then $n$-restricted QFA $M$ with probability at least $p$ only after the entire input has been read, the probability of observing $E_{acc}$ if $a_i = 0$, or $E_{rej}$ if $a_i = 1$, is at least $p$. Thus, $f$ defines a serial encoding, as claimed. □

It then follows from Theorem 2 that $\lceil \log |Q| \rceil = \Omega(\frac{k}{\log k})$, but $k = \frac{n}{2}$, hence $|Q| = 2^{\Omega(\frac{n}{\log n})}$. From Theorem 3 it follows that any quantum automaton that recognizes $L_n$ also requires $2^{\Omega(\frac{n}{\log n})}$ states.

3. Easy.

4. The PFA $Q$ in Theorem 1 has one rejecting ($q_{rej}$), one accepting ($q_{acc}$), one initial ($q_{ini}$) state, and many non-halting states $q_i$. We build PFA $Q'$ recognizing language $L_n$ with one rejecting ($q'_{rej}$), one accepting($q'_{acc}$), one starting ($q'_{ini}$) state, and several non-halting states $q'_{i,0}$, $q'_{i,1}$ and $q'_{i,2}$, where $i$ is an index of a state of $Q$. For every transition from state $q_i$ to state $q_j$ with probability $p$ for

the input symbol $a$ (we denote this by $f(q_i, a, q_j, p)$) there are six transitions in $Q'$ (denoted by $f'$):

1. $f'(q'_{i,0}, 1, q'_{i,0}, p)$
2. $f'(q'_{i,0}, 0, q'_{i,1}, p)$
3. $f'(q'_{i,1}, 1, q'_{i,0}, p)$
4. $f'(q'_{i,1}, 0, q'_{i,2}, p)$
5. $f'(q'_{i,2}, 1, q'_{i,2}, p)$
6. $f'(q'_{i,2}, 0, q'_{i,2}, p)$

For every $f(q_{ini}, \emptyset, q_i, p)$, there is $f'(q'_{ini}, \emptyset, q'_{i,0}, p)$; for every $f(q_i, a, q_{rej}, p)$ there are $f'(q'_{i,k}, x, q'_{rej}, p)$, $k \in \{0, 1, 2\}$, $x \in \{0, 1\}$, and for every $f(q_i, \$, q_{rej}, p)$ there are $f'(q'_{i,k}, \$, q'_{rej}, p)$, $k \in \{0, 1, 2\}$; and for any $f(q_i, \$, q_{acc}, p)$ there are $f'(q'_{i,2}, \$, q'_{acc}, p)$, $f'(q'_{i,0}, \$, q'_{rej}, p)$, $f'(q'_{i,1}, \$, q'_{rej}, p)$.

Informally, we make three copies of states in $Q$ and their meaning is similar to those of the automaton of Lemma 1. The automata answer in parallel two questions: is the length of an input word $n$?, and, are there any adjacent zeroes in it? It is obvious that the accepted words are those with both answers 'yes'. □

## 4 Conclusion

We have shown that quantum automata must be sometimes almost doubly exponentially larger than equivalent classical automata. A related question, however, remains. As shown by Ambainis and Freivalds [2], any language accepted by a QFA with high enough probability can be accepted by a RFA which is at most exponentially larger that a minimal DFA accepting the language. It thus follows that Theorem 4 is close to maximal gap between probabilistic and quantum automaton with high enough probability of success (as precisely computed by Ambainis and Ķikusts [8] - greater than $\frac{52+4\sqrt{7}}{81} = 0.7726...$) But the situation is not clear when we allow smaller probability of correctness. The author does not now of any lower or upper bound in this case.

**Acknowledgement**. I would like to thank Rūsiņš Freivalds for suggesting the problem and for his help during research.

## References

1. A. Kondacs and J. Watrous. *On the power of quantum finite state automata.* Proc. 38th FOCS, 1997, pp. 66-75.
2. A. Ambainis and R. Freivalds. *1-way quantum finite automata: strengths, weaknesses, and generalizations.* Proc. 39th FOCS, 1998, pp. 332-341.
3. A. Ambainis, A. Nayak, A. Ta-Sham and U. Vazirani. *Dense Quantum Coding and a Lower Bound for 1-way Quantum Automata.* http://xxx.lanl.gov/ps/quant-ph/9804043, 1998.
4. Z. Rasščevskis. *The Complexity of Probabilistic versus Deterministic Finite Automata.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning, Proc. Int. Workshop, Sweden, 27-29 May 2000, pp. 89-92. http://www.ima.mdh.se/personal/rbr/courses/sundbyholmproc/zigmars.ps

5. R. Freivalds. *On the growth of the number of states of determinization of probabilistic automata.* Avtomatika i Viscislitelnaja Tehnika, 1982, N.3, pp. 39-42. (in Russian)
6. M. O. Rabin. *Probabilistic Automata.* Information and Control, 6(1963), pp. 230-245.
7. A. Ambainis. *The complexity of probabilistic versus deterministic finite automata.* Proc. ISAAC'96, Lecture Notes in Computer Science 1178, 1996, pp. 233-237.
8. A. Ambainis and A. Ķikusts. *Exact results for accepting probabilities of quantum automata.* quant-ph/0109136, 2001.

# Probabilistic and Quantum Automata for Undecidability Proofs

Gints Tervits*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia

**Abstract.** We use ideas from [Fr78] to construct a 2-tape quantum finite automaton for a language that was supposed to be crucially important to prove undecidability of the emptiness problem for 2-tape quantum finite automata recognizing languages with fixed probability of the correct result exceeding $\frac{2}{3}$.

## 1 Introduction

A rather interesting effect was recently discovered. Bonner, Freivalds and Gailis [BFG00] proved that for 1-way 2-tape quantum finite automata (2-QFA) recognizing a language with a known probability $\frac{2}{3}$ of the correct result it is undecidable whether the recognizable language is empty. However it is unknown whether the same emptiness problem remains undecidable in the case if the probability of the correctness exceeds $\frac{2}{3}$.

Bonner, Freivalds and Rikards [BFR00] proved that for 1-way finite automata with a counter recognizing a language with a known probability $\frac{2}{3}$ of the correct result it is undecidable whether the recognizable language is empty. However it is unknown whether the same emptiness problem remains undecidable in the case if the probability of the correctness exceeds $\frac{2}{3}$.

Freivalds and Winter [FW01] proved that for 1-way finite state transducers computing an input-output relation with a known probability $\frac{2}{3}$ of the correct result it is undecidable whether the recognizable relation is empty. However it is unknown whether the same emptiness problem remains undecidable in the case if the probability of the correctness exceeds $\frac{2}{3}$.

We have not succeeded to solve any of these open problems. However we have discovered that in the case of 1-way 2-tape finite automata a language that was considered crucially important for proving the undecidability result for probabilities exceeding $\frac{2}{3}$ is recognizable by probabilistic and quantum 2-tape finite automata with arbitrarily high probability $1 - \epsilon$.

* Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission, and the Swedish Institute, Project ML-2000

## 2 Lemma for joining unlimited number of counters

Let $N$ denote the set of all non-negative integers, $Z$ denote the set of all integers and let $X$ denote an arbitrary set. Let $n \in N$. Let P be a function $X \to \{0,1\}$ and $F$ be a function $X * \{1,2,\ldots,n\} \to Z$. We call the pair of functions $\langle P, F\rangle$ dispersive if for all $x \in X$ the following holds:

1. $P(x) =\supset (\forall u, v \in \{1,2,\ldots,n\})(F(x,u) = F(x,v))$.
2. $P(x) =\supset (\forall u, v \in \{1,2,\ldots,n\})(u \neq v \supset (F(x,u) \neq F(x,v))$.

Let $n \in N$ and $k \in N$ and for every $I \in \{1,2,\ldots,k\}$ a disperse pair of functions $(P_i : X \to \{0,1\}; F_i : X * \{1,2,\ldots,n\} \to Z)$. We denote the family $\{F_1, F_2, \ldots, F_n\}$ by $F$. We consider the following random value $S_f(x)$. For arbitrary $I \in \{1,2,\ldots,k\}$ a random number $Y_i$ is taken which is distributed equiprobably in $\{1,2,\ldots,n\}$ and every $Y_i$ is statistically independent from other $Y_j$. Then

$$S_f(x) = \sum_i^k F_i(x, y_i).$$

**Lemma 1.** [Freivalds [Fr78]] *Let $n \in N$ and $k \in N$ and for every $i \in \{1,2,\ldots,k\}$ a disperse pair of functions $(P_i : X \to \{0,1\}; F_i : X * \{1,2,\ldots,n\} \to Z)$. Let $F = \{F_1, F_2, \ldots, F_n\}$. Then for arbitrary $x \in X$ if $\prod_i^k P_i(x) = 1$ there is $z \in Z$ such that $S_f(x) = z$ with probability 1 and if $\prod_{i=1}^k P_i(x) = 0$ there is not a single $z$ such that the probability of $S_f(x) = z$ would exceed $\frac{1}{2}$.*

*Proof.* The first assertion is evident. To prove the second assertion we follow [Fr78]. We assume that $i \in \{1,2,\ldots,k\}$ is hold $P_i(x) = 0$. Then, by the second property of a dispersive pair of functions, the values $F_i(x,1)$, $F_i(x,2)$, ..., $F_i(x,n)$, are all different. Random value $S_f(x)$ is denoted as the sum $\sum_{j=1}^k F_j(x, y_j)$ including $F_i(x, y_i)$ and every $Y_i$ is statistically independent from al other $Y_j$. But for arbitrary $z \in Z$ and arbitrary set

$$\{F_1(x, y_1), \ldots, F_{i-1}(x, y_{i-1}, F_{i+1}(x, y_{i+1}, \ldots, F_k(x, y_k)\}$$

the sum $\sum_{j=1}^k F_j(x, y_j)$ can equal $z$ no more than one of the $n$ possible values $Y_i$. □

## 3 Possibilities of multi-tape finite automata

We define the language $C^2$ consisting of all possible pairs of words

$$(0^{l(k-1)}, \{\underbrace{0^l 1 0^l 1 0^l 1 \ldots 1 0^l 2 0^l}_{k \text{ times}}), \}$$

type, where $l \geq 1, k \geq 2$. (The second word in the pair contains $k-2$ entries of letter 1 and $k$ entries of subword $0^l$).

**Theorem 1.** *(1) For arbitrary $\epsilon > 0$ there is a 2-tape probabilistic finite automaton recognizing $C^2$ such that it accepts every pair of words in $C^2$ with probability 1 and rejects every pair of words in $\neg C^2$ with probability $1 - \epsilon$. (2) $C^2$ does not belong to the Boolean closure of languages recognizable by 2-tape deterministic finite automata.*

*Proof.* (1) Let $n$ be the non-negative integer $\frac{1}{n} < \epsilon$. The automaton can easily test if the pair of words belong to language

$$D^2 = \{(0^s, 0^{l_1}10^{l_2}1 \dots 10^{l_{k-1}}20^{l_k}) \mid s, k, l_1, l_2, \dots, l_k \in N\}.$$

Provided $X \in D^2$, the automaton performs a calculation defined by the following family of dispersive pairs of functions $\{(P_i, F_i)\}$, where $i \in \{1, 2, \dots, k-1\}$.

$$P_i(x) = 1 \text{ if } l_i = l_{i+1}, \text{ and } P_i(x) = 0 \text{ if } (m, n) = 1.$$

$$F_i(x, y) = \frac{l_i * y + l_{i+1} * (n + 1 = y)}{n + 1}.$$

Let $F = (F_1, f_2, \dots, F_n)$. Then

$$S_f(x) = \sum_{i=1}^{k-1} \frac{l_i * y_i + l_{i+1}(n + 1 - y_i)}{n + 1}.$$

The probabilistic finite automaton processing pairs of words

$$X = (0^s, 0^{l_1}10^{l_2}1 \dots 10^{l_{k-1}}20^{l_k})$$

gradually takes random numbers $y_1, y_2, \dots, y_{k-1}$ (no more than two at a time) with equal probability and being independent in set $\{1, 2, \dots, n\}$ and does next calculations. While second head goes through $0^{l_1}$ for distance of $n + 1$ cells, the first head goes through $y + i$ cells. The first head goes through $\frac{l_1 y_1}{n+1}$ cells all the time. While the second head goes through $0^{l_j}$ where $2 \le j \le k - 1$, at moment it reads $n + 1$ cells the first head goes through $(n + 1 - y_{j-1} + y_j$ cells. The first head goes through $\frac{l_j * (n+1-y_{j-1}}{n+1} + \frac{l_j * y_j}{n+1}$ cells in all the time. While the second head goes through $0^{l_k}$, at moment it reads $n + 1$ cell the first head goes through $(n + 1 - y_{k-1})$ cells. In all time the first head goes through $\frac{l_j * (n+1-y_{j-1}}{n+1}$ cells. If the first head has finished going through word $0^s$ at the same time the second head has finished the word, if $S = S_f(x)$ the automaton accepts pair $x$, otherwise it is rejected. Since every pair of functions from family $\{(p_i, F_i)\}$ is disperse we use Lemma 1 to provide required possibility of the automaton.

(2) Assume the contrary. We denote language $c^2$ by $f(L_1^{(2)}, \dots, L_n^{(2)})$ where f - Boolean operation and by $L_1^{(2)}, \dots, L_n^{(2)}$ languages of pairs of the words recognized by two-head deterministic finite automaton $M_1, \dots, M_n$. We denote the number of inside position of these automata by $a_1, \dots, a_n$ and $max\{a_1, \dots, a_n\}$ by $a$. We note that the first head actually can not read but only finds out if

it has reached the end of the word. We denote $a^n + 3$ by $k$, $3a^n + 1$ by $l$ and consider how automata $M_1, \ldots, M_n$ process pair of words

$$X = (0^{l*(k-1)}, 0^l 1 \ldots 10^l 20^l),$$

belonging to language $C^2$. We consider the internal state of automata $M_1, \ldots, M_n$ when the second head is reading first letter of each set of zeros. Each set of zeros we compare to vector of distances $(q_1, \ldots, q_n)$. Number of zero sets is $a^n + 3$, number of any probable vectors do not exceed $a^n$. So there are two such sets of zeros no one being first or last and all automata $M_1, \ldots, M_n$ start going through these sets at the same internal state. It follows that the second, the third, etc. letters of both sets of zeros will be read by automata in equal states. We call these sets of zeros marked. Since the number of se of zeros is large comparing with number of states, the states will be repeated recurrently, but we must say, with different periods. As the number of automata is finite, there is the least multiple $p$ of these periods. Value $p$ is common period for all the automata and do not exceed $a^n$. We consider work of automata $M_1, \ldots, M_n$ with pair of words

$$y = 3D(0^{l*(k-1)}, 0^l 1 \ldots 10^l 10^{l-p} 10^l 1 \ldots 10^l 1 = 0^{l+p} 10^l 1 \ldots 10^l 20^l),$$

differing from pair $x$. The first set is shortened for $p$, second is extended for $p$. Pair $y \notin C^2$, but no automata $M_1, \ldots, M_n$ distinguish pair $y$ from pair $x$. When the reading of the word by second head is ended it has the same internal state and the first head has covered the same distance. It follows that $y$ and $x$ belongs to the same languages $L_1^{(2)}, \ldots, L_n^{(2)}$. So $C^2 \neq f(L_1^{(2)}, \ldots, L_n^{(2)})$. Contradiction. □

**Theorem 2.** *(1) For arbitrary $\epsilon > 0$ there is a 2-tape quantum finite automaton recognizing $C^2$ such that it accepts every pair of words in $C^2$ with probability 1 and rejects every pair of words in $\neg C^2$ with probability $1 - \epsilon$. (2) $C^2$ does not belong to the Boolean closure of languages recognizable by 2-tape deterministic finite automata.*

*Proof.* (1) This part of the proof simulates the corresponding part of the proof of Theorem 1. However the random branching at the needed places is substituted by Quantum Fourier Transform. For prime values of the parameter $p$, QFT is defined by the following matrix.

$$\begin{pmatrix}
\frac{1}{\sqrt{p}}(e^0) & \frac{1}{\sqrt{p}}(e^0) & \frac{1}{\sqrt{p}}(e^0) \ldots & \frac{1}{\sqrt{p}}(e^0) \\
\frac{1}{\sqrt{p}}(e^{\frac{2p\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{2(p-1)\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{2(p-2)\pi}{p}}) \ldots & \frac{1}{\sqrt{p}}(e^{\frac{2\pi}{p}}) \\
\frac{1}{\sqrt{p}}(e^{\frac{4p\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{4(p-1)\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{4(p-2)\pi}{p}}) \ldots & \frac{1}{\sqrt{p}}(e^{\frac{4\pi}{p}}) \\
\frac{1}{\sqrt{p}}(e^{\frac{6p\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{6(p-1)\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{6(p-2)\pi}{p}}) \ldots & \frac{1}{\sqrt{p}}(e^{\frac{6\pi}{p}}) \\
\ldots & \ldots & \ldots \ldots & \ldots \\
\frac{1}{\sqrt{p}}(e^{\frac{(p-1)p\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{(p-1)(p-1)\pi}{p}}) & \frac{1}{\sqrt{p}}(e^{\frac{(p-1)(p-2)\pi}{p}}) \ldots & \frac{1}{\sqrt{p}}(e^{\frac{(p-1)\pi}{p}})
\end{pmatrix}$$

If the pair of words is in the language, the automaton reaches an accepting state on all the computation paths. If the pair of words is not in the language, the automaton rejects it on *most* of computation paths. □

## References

[BFG00] R. Bonner, R. Freivalds and R. Gailis. *Undecidability of 2-tape Quantum Finite Automata.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning. Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 93-100.

[BFR00] R. Bonner, R. Freivalds and M. Rikards. *Undecidability on Quantum Finite 1-Counter Automaton.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning. Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 65-71.

[FW01] R. Freivalds and A. Winter. *Quantum finite state transducers.* Lecture Notes in Computer Science 2234, Springer, 2001, pp.233-242.

[Fr78] R. Freivalds. *Recognition of languages with high probability by various types of automata.* Dokladi AN SSSR, 1978, v. 239, No. 1, pp. 60-62 (in Russian)

# Quantum Automata with Mixed States: Function Computation and Language Recognition

Andrej Dubrovsky* and Oksana Scegulnaja**

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
oksana.s@liis.lv

**Abstract.** We consider two computational models, pure states quantum automata and quantum automata with mixed states, and show that the latter can recognize certain languages better than the former. We also compare the two types of automata with respect to the complexity of Boolean function computation.

## 1 Introduction

The computational power of the quantum mechanism is often superior to both deterministic and probabilistic classical models. There exists a quantum algorithm, introduced by Deutsch and Jozsa [7], which computes XOR function with two arguments making only one oracle call. Grover's quantum algorithm [8] for database search requires $O(\sqrt{n})$ oracle calls, as compared to O(n) calls required in deterministic search; the algorithm yields the same complexity of computation with bounded error for AND and OR functions of $n$ arguments. In the presence of promises about the input, some quantum algorithms can achieve exponential speed-up over the classical ones, for example, Simon's algorithm [13]. Without any promises, however, as shown by Beals et al [5], the gap between quantum and classical (that is, deterministic or probabilistic) algorithms is at most polynomial.

In this mote, we consider two computational models: quantum automata with pure states, and with mixed states. The model with mixed states allows a quantum system to be in several superpositions of states with some probability. Using this model, Ambainis [1, 2] improved the lower bounds for query algorithms.

We produce a Boolean function of six binary variables, which can be computed by quantum automata with mixed states using two oracle calls. For this function, Ambainis' bounds yield at least $\Omega(\sqrt{6})$ calls for pure states automata, an inconclusive bound.

* Research supported by Grant No.01.0354 from the Latvian Council of Science, and Contract IST-1999-11234 (QAIP) from the European Commission.

** Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission, and grant from A/S Dati Grupa.

We also show that the language the language $a^+$ over a two letter alphabet, consisting of all words of form $a^n$, $n > 0$, recognizable by pure state 1-way finite quantum automata with probability at most $\frac{52+4\sqrt{7}}{81} = 0.7726\ldots$ [4], can be recognized by Najak's [11] mixed state automata with probability arbitrarily close to one.

## 2 Preliminaries

We refer to Gruska [9] for basic facts on quantum computation. Here, we consider the complexity of computation of a Boolean function $f : \{0,1\}^N \longrightarrow \{0,1\}$ in the (black-box) quantum query model, as described in [5] and [1]. In this model, a quantum computation with $T$ queries is a sequence

$$U_0 \longrightarrow O \longrightarrow U_1 \longrightarrow O \longrightarrow \ldots \longrightarrow U_{T-1} \longrightarrow O \longrightarrow U_T$$

of unitary transformations, with $U_i$ arbitrary and independent of the input values, and $O$ is a query transformation. There are two natural and equivalent ways of defining $O$. We may let $O$ perform the XOR operation on the $i$-th bit queried: $|i,b,z\rangle \longrightarrow |i,b \oplus x_i,z\rangle$, where $1 \leq i \leq \lceil logN \rceil$, $b$ is a working bit, and $z$ stands for all the other bits. Alternatively, we may change phase depending on $x_i$: $|i,b,z\rangle$ to $(-1)^{b*x_i}|i,b,z\rangle$. The computation starts in some state $|q_0\rangle$, the transformations $U_0, O, U_1, \ldots O, U_T$ are applied, and the final state is measured, outputting the rightmost bit as the value of the function $f$. The algorithm computes $f$ with bounded error, if for every input value $x$, the probability of correct answer $f(x)$ is at least $1-\varepsilon$ for some fixed $\varepsilon < \frac{1}{2}$. It is usual to take $\varepsilon = \frac{1}{3}$.

There are several methods for estimating lower bounds for quantum algorithms. Most popular are the classical adversary method [6, 12], which tries to alternate the input without affecting the output; the method of using polynomials [5]; and quantum adversary method [1, 2], that runs on superposition of inputs and estimates the number of queries needed to achieve entanglement between the algorithm and the oracle work spaces.

## 3 Results

### 3.1 Function computation

Let $f$ be the Boolean function of six binary variables $x_i$, $1 \leq i \leq 6$, which equals one if and only if $x_1x_2 = x_3x_4 = x_5x_6$ (equality of words).

**Theorem 1.** *There is a quantum automaton with mixed states, which computes $f$ with bounded error using exactly two oracle queries.*

*Proof.* Consider a quantum automaton with mixed states, working as follows. It starts in three superpositions of states $q_{10}, q_{20}$ and $q_{30}$ with probability $\frac{1}{3}$ each. Then, each of the three branches computes a part of the function corresponding to the three conditions $x_1x_2 = x_3x_4$, $x_3x_4 = x_5x_6$, and $x_1x_2 = x_5x_6$. Examine,

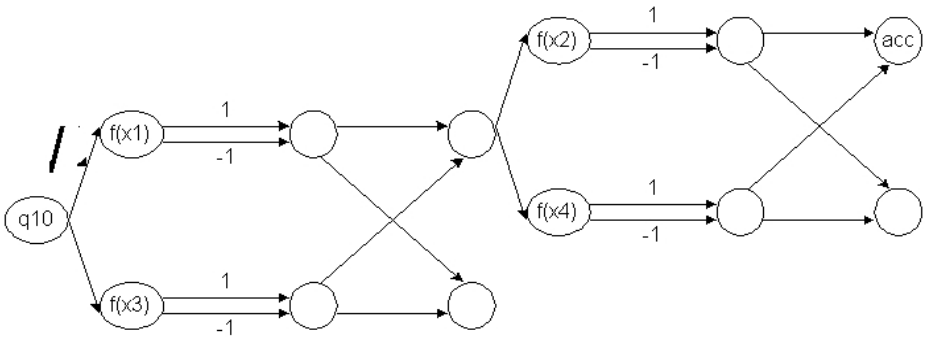


**Fig. 1.** The work of the first branch

for example, the work of the first branch, depicted as a graph in Figure 1. In this graph, all unmarked edges carry the amplitude $1/\sqrt{2}$ with the exception of the two lowermost edges where the amplitude is reversed to $-1/\sqrt{2}$.

We check the condition $x_1x_2 = x_3x_4$ by computing the function $x_1 \oplus x_3 \vee x_2 \oplus x_4$ containing two XOR operations. Consequently, the branch consists of two XOR operations, the second one linked to the accepting state of the first one. Since a XOR operation can be processed with a single oracle call, see [7], making two XOR operations one after another requires two oracle.

All in all, a word $x_1x_2 \dots x_6$ satisfying the defining conditions for $f$ is accepted with probability 1, otherwise it is rejected with probability $\frac{2}{3}$. □

It would be interesting to know the lower bound on the number of queries used by a pure state quantum automaton computing our function $f$. Using Ambainis' results [1, 2], and specifically Theorem 2 in [1], we only obtain a bound in the inconclusive form $\Omega(\sqrt{6})$.

### 3.2 Language recognition

We turn to the question whether allowing mixed states in 1-way quantum finite automata (QFA) may result in a higher probability of language recognition, than when using pure states only.

1-way QFA with pure states were studied by Kondacs and Watrous [10]. It is rather simple and very limited in its computational power: it only recognizes a proper subclass of the regular languages. One of the non-recognizable languages is $\{0,1\}^*1$. For some other languages there exist proven upper bounds for recognition probabilities; for example, for the language $a^+$ over a two letter alphabet, consisting of all words of form $a^n$, $n > 0$, the (sharp) upper bound is $\frac{52+4\sqrt{7}}{81} = 0.7726\dots$ [4].

1-way QFA with mixed states (QFAMS) were introduced by Najak [11] as *enhanced* QFA. They allow not only the collective measurements of the accepting, rejecting, and non-halting sets of states, but also individual measurements of each non-halting state. The measurements are performed after every letter read.

**Theorem 2.** *For any $\epsilon > 0$ there exist a 1-way QFAMS (enhanced QFA) which recognizes the language $a^+$ with probability $1 - \epsilon$.*

*Proof.* Consider an automaton $A$ with $n$ working states $q_0, \ldots, q_{n-1}$, of which $q_0$ is a starting state; $n-1$ accepting states $qa_1, \ldots, qa_{n-1}$; and $n$ rejecting states $qr_0, \ldots, qr_{n-1}$. In total, there are $n+2$ measurement operators: one for each of the non-halting states, one for the set of accepting states, and one for the set of rejecting states.

The transition matrices of $A$: $U_a$, $U_b$, and $U_\$$, corresponding to the letters $a$, $b$, and the right delimiter \$, are defined as follows. $U_a$ operates as the quantum Fourier transform [9] in the subspace of working states; in particular, any 'point' superposition centered at a working state is mapped by $U_a$ into a uniform superposition of all working states. $U_b$ maps all the working states to rejecting states. $U_\$$ maps the initial state $q_0$ to a rejecting state, and all the other $n-1$ working states to respective accepting states.

The automaton works as follows. When started, it is in state $q_0$. If, at this moment, the $b$ or \$ is read from the tape, the automaton rejects with probability 1; if $a$ is read, the operator $U_a$ is applied and the resulting superposition is measured using all the $n+2$ measurement operators. After the measurement, the automaton is any of the working states with probability $\frac{1}{n}$. The reading of subsequent $a$:s does not alter the state distribution, because states collapse on non-halting measurements, and the quantum Fourier transform re-distributes them uniformly. So, when \$ is finally read, the automaton accepts with probability $1 - \frac{1}{n}$, which is the probability that the state is $q_0$. Obviously, $\frac{1}{n}$ can be made arbitrary small by taking the number $n$ of states of the automaton large enough. □

## 4 Conclusion

We have looked for advantages of mixed states automata over pure states automata. We have considered a function that can be computed by both types of automata with comparable query complexity. We have also presented a mixed states automaton that violates the upper bound for language recognition probabilities for pure states automata, showing that the mixed states model is in some cases more powerful.

## References

1. A. Ambainis. *Quantum lower bounds by quantum arguments.*, quant-ph/0002066, 2000.

2. A. Ambainis. *Quantum query model: algorithms and lower bounds.* In: Proc. Second International Workshop on Quantum Computing and Learning, 1999
3. A. Ambainis and R. Freivalds. *1-way Quantum finite automata: strengths, weaknesses and generalizations.* Proc. FOCS'98, pp. 332-341.
4. A. Ambainis and A. Kikusts. *Exact results for accepting probabilities of quantum automata.*
5. R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. *Quantum lower bounds by polynomials.* Proc. FOCS'98, pp. 351-361. Also arXiv:quant-ph/9802049v3.
6. C. Bennett, E. Bernstein, C. Brassard and U. Vasirani. *Strengths and weaknesses of quantum computing.* SIAM Journal on Computing, 26(3), 1997, pp. 1510-1523. Also quant-ph/9701001.
7. D. Deutsch and R. Jozsa, R. *Rapid solution of problems by quantum computation.* Proc. Royal Society, London, vol. A439, 1992, pp. 553-558.
8. L. K. Grover. *A fast quantum mechanical algorithm for database search.* Proc. 28th STOC, 1996, pp. 212-219.
9. J. Gruska. *Quantum computing,* McGraw Hill, 1999.
10. A. Kondacs and J. Watrous. *On the power of Quantum State Finite Automata.* Proc. 38th Symp. on Foundations of Computer Science, 1999, pp. 66-75.
11. A. Najak. *Optimal lower bounds for quantum automata and random access codes.*
12. Y. Shi. *Lower bounds of quantum black - box complexity and degree of approximation polynomials by influence of Boolean variables.* Information Processing Letters, 75, 2000, pp. 79-83. Also quant-ph/9904107.
13. D. Simon. *On the power of quantum computation.* Proc. 35th IEEE Symp. on Foundations of Computer Science, 1994, pp. 124-134.

# Query Automata Minimization

Dmitry Kuzmenko*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
morfizm@yahoo.com

**Abstract.** We introduce algorithms which search for the best randomized (probabilistic) and quantum query automata for a given Boolean function. Some results obtained with these algorithms, are reported. The main idea is to construct a general form of an automaton with a bound on the number of queries, where all parameters are variable, and then to apply an optimization algorithm to find the automaton that evaluates a given function with best probability.

## 1 Randomized query automata

### 1.1 The structure

We consider randomized query automata (RQA) computing Boolean functions of $n$ binary variables $x_1, \ldots, x_n$. If the number of queries in the computation is a priori bounded by an integer $N$, every such automaton is naturally represented in a standard form of a tree $T$ of depth $2N + 2$, as follows. The root of $T$ is a branching node: it links, with specified probability weights, to all the admissible query nodes $x_k$, $1 \leq k \leq n$. Each of these query nodes links in turn to two new branching nodes, the links weighted by the outcome of the query (0 or 1). In turn, each of the new branching nodes links, with specified probability weights, to all query nodes not present in its backward path to the root. After $N$ steps of this construction, the terminal branching nodes each link to two leaves of the tree, labelled 0 or 1, representing the outcome of a computation.

On input $x$, the automaton follows a path through the tree $T$: it randomly selects the first query, conditionally on its outcome it randomly selects the second query, etc, until all $N$ queries have been executed, after which it randomly outputs 0 or 1. Thus, every path through $T$ is a sequence of alternating queries and random branching ('coin flipping'). For example, for $n = 4$ arguments and $N = 2$ queries, there are $4 \times 2 \times 3 \times 2 \times 2 = 96$ different computational paths in the tree.

Thus, the search for a RQA for computing a given function in a bounded number of queries is reduced to the search for the probability weights in the tree which represents it.

* Research supported by contract IST–1999–11234 (QAIP) from the European Commision and grant no. 01.0354 from the Latvian Council of Science.

### 1.2 How to seek the probabilities

The large number of variables suggests random search. The simplest approach is to start from some initial automaton and iteratively making small random changes in order to increase its *value*, here defined as the probability of correct answer in the worst case of input data. There are natural choices of the initial automaton: a *random* choice (all probabilities are chosen randomly), a *symmetric automaton* (all probabilities are equal), or some good automaton found manually.

Note that the optimization is *constrained* - the sum of all probabilities in a branching must remain equal to one - so any change violating this condition must be followed by normalization (changes respecting the constraints are rotations of branching probabilities by small angles; these may be easily implemented for pairs of probabilities, but there is an extra computational cost).

The main problem of this form of random search, as it is well known, is the possible existence of multiple local maxima of the value function. If we allow only 'good' changes, which instantly increase the value of RQA, we are likely to end up with a local but not global maximum. To address this problem, it is customary to keep some 'bad' changes as well, as it is done, for example, in genetic programming.

### 1.3 Genetic approach

Consider an RQA as an *individual*, a collection of n RQA - as a *population*. The next *generation* of the population is created by *crossing* some individuals and *mutating* some of the resulting *children* - of all c children, only n are selected into the next generation. The ways of crossing, mutating and selecting are ambiguous.

Crossing two parents might, for example, consist in swapping two of their randomly chosen subtrees - the *genes*; naturally, by the fixed tree structure of all RQA under consideration, the swapped subtrees must be of equal depth.

A mutation need not be a small change. We may mutate a tree node, but we may also delete subtrees (genes) and generate them anew. All mutations should not be of equal probability: the 'heavy' mutations, for example, which completely rebuild a RQA, should occur relatively rarely.

To select a new generation, a suitable algorithm is 'a tournament'. We select $t$ RQA from $c$ candidate children, and choose the best among them. This is being repeated until we collect all $n$ RQA for the next generation. If we choose $t$ close to $c$, this will be almost the same as selecting $n$ best RQA (there is then a risk of degeneration - of getting struck in a local extremum). If $t = 1$, then choice is completely random, and there are few chances to improve the population. So, the value of $t$ should be chosen appropriately; if $n = 100$, for example, a suitable $t$ may be of the order 10.

*Results* This method suggests that the function

$$f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4) \vee (x_4 \wedge x_1) \qquad (1)$$

cannot be computed by RQA with two queries and probability higher than $\frac{3}{4}$. However, negative results achieved in this way should be interpreted with caution. The chance that $\frac{3}{4}$ may be just a local extremum is minimized, but not excluded.

## 2 Quantum query automata

### 2.1 The structure

Quantum query automata (QQA) are defined similarly to RQA, but with probabilities replaced by complex amplitudes. We recall a universal definition [25]: 'A quantum computation with $T$ queries is just a sequence of unitary transformations

$$U_0 \to O \to U_1 \to O \to \ldots \to U_{T-1} \to O \to U_T \tag{2}$$

The $U_j$ 's can be arbitrary unitary transformations that do not depend on the input bits $x_1, \ldots, x_n$. $O$ are query (oracle) transformations. To define $O$, we represent basis states as $|i, b, z\rangle$, where $i$ consists of $\log N$ bits, $b$ is one bit, and $z$ consists of all other bits. Then, $O$ maps $|i, b, z\rangle$ to $|i, b \oplus x_i, z\rangle$ (i.e., the first $\log N$ bits are interpreted as an index $i$ for an input bit $x_i$ and this input bit is XOR-ed on the next qubit). We use $O_x$ to denote the query transformation corresponding to an input $x = (x_1, \ldots, x_n)$." As the 'tree structure' is not fixed here, we must limit the number of states, so the size of the matrices representing the operators $U$ and $O$ is pre-defined. It must thus be pre-calculated to be large enough to allow to find the automaton, but not too large for efficiency reasons. We need at least $(2n)T$ states.

There is an alternative definition of QQA due to Ambainis [4]: ' ... Also, we can define that $O$ maps $|i, b, z\rangle$ to $(-1)^{bx}|i, b, z\rangle$ (i.e., instead of XOR-ing $x_i$ on an extra qubit, we change phase depending on $x_i$). It is well known that both definitions are equivalent ...'. This definition reduces the number of states required, as the operation of changing phase requires additional state when simulated by XOR-ing.

### 2.2 The algorithm

We use the same technique as for RQA, only instead of subtrees, we take as genes the individual matrixes from (2), or all matrixes in (2) to the right of a selected matrix. Mutations will differ for matrixes $U$ and $O$. In the first case, mutations are unitary transformations on randomly selected matrix $U$. It is also possible to mutate not an entire matrix, but only a part of it, for example, by multiplying it with a $2 \times 2$ unitary matrix tensored with an identity matrix; this mutation will change only two columns in $U$. $2 \times 2$ unitary matrixes can be generated in different ways, for example, by customizing the following patterns:

$$\begin{pmatrix} \cos\phi\ e^{i\alpha} & -\sin\phi\ e^{i\beta} \\ \sin\phi\ e^{i\chi} & \cos\phi\ e^{i\delta} \end{pmatrix} \tag{3}$$

where $\alpha + \delta = \beta + \chi$, or,

$$\begin{pmatrix} \cos\phi\ e^{i\alpha+\chi} & -\sin\phi\ e^{i\alpha+\delta} \\ \sin\phi\ e^{i\beta+\chi} & \cos\phi\ e^{i\beta+\delta} \end{pmatrix} \tag{4}$$

with $\alpha, \beta, \delta, \chi$ arbitrary.

Larger mutations are also needed, but a 'big' unitary matrix can be obtained by combining several $2\times2$ matrixes. Sine and cosine are fast to compute for small angles, if pre-calculated (reasonable amount of computer memory is enough to store a table for quick calculating of $\sin\alpha$ and $\cos\alpha$ for $0 < \alpha < 5^\circ$ with, say, 16 decimal digits of precision. After many mutations, matrices may lose unitarity due to accumulation of errors. This problem can be fixed in two ways: first, after multiplying matrixes, we can store not only the result, but also both source matrixes, to be used in subsequent multiplications. Second, we can unitarize matrixes from time to time; unitarization is orthogonalization of the column space and normalization, and can be done by the usual Gram-Schmidt procedure. In spite of this process having complexity $O(n^3)$, it is affordable if not done too often.

*Results* One of the results is that function (1) can be computed by QQA with one query and probability of correct answer at least 0.637539.

## References


1. A. Ambainis. *Quantum lower bounds by quantum arguments.* 2000. http://citeseer.nj.nec.com/432640.html, http://www.cs.berkeley.edu/ ambainis/ps/density4.ps.gz
2. A. Ambainis. *Quantum Query Model: Algorithms and Lower Bounds.* 1999. http://www.ima.mdh.se/personal/rbr/courses/riga99proc.html, http://www.ima.mdh.se/personal/rbr/courses/riga99proc/ambainis.ps
3. R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. *Quantum lower bounds by polynomials.* Proc. FOCS'98, pp. 351-361. http://citeseer.nj.nec.com/51642.html, http://www.cwi.nl/ rdewolf/publ/qc/focs98.ps.gz
4. A. Berzina, R. Bonner and R. Freivalds. *Parameters in Ambainis-Freivalds algorithm.* In: R. Bonner and R. Freivalds, Quantum Computation and Learning, Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 101 - 109.
5. H. Buhrman, R. Cleve, R. de Wolf and C. Zalka. *Bounds for small-error and zero-error quantum algorithms.* 1998. http://xxx.lanl.gov/abs/quant-ph/9802062, http://citeseer.nj.nec.com/198249.html, http://www.cwi.nl/ rdewolf/publ/qc/focs99.ps.gz
6. H. Buhrman and R. de Wolf. *Complexity Measures and Decision Tree Complexity: A Survey.* 2000. http://citeseer.nj.nec.com/buhrman00complexity.html, http://www.cwi.nl/ rdewolf/publ/qc/dectree.ps.gz
7. R. Freivalds and L. Lace. *Better upper bounds for advantages of probabilistic and quantum decision trees.* Unpublished manuscript, 2002.

# Generation of Quantum Finite Automata by Genetic Programming

Alexey Luchko

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
sd90003@latnet.lv

**Abstract.** I discuss the idea of using genetic programming for building a quantum finite automaton (QFA) for a predefined language. The problems to solve include: describing a QFA population, defining genetic operations on QFA, and, finding a fitness function.

## 1 Quantum finite automata

A one-way quantum finite automaton (QFA) $A$ is specified by a finite (input) alphabet $\Sigma$, a finite set $Q$ of states, an initial state $q_0 \in Q$, two disjoint sets $Q_a \subseteq Q$ and $Q_r \subseteq Q$ of *accepting* and *rejecting* states[1], respectively, and a transition function $\delta : Q \times \Gamma \times Q \to C_{[0,1]}$, where $\Gamma = \Sigma \cup \{\#, \$\}$ is the tape alphabet of $A$, and $\#$ and $\$$ are left and right markers not in $\Sigma$. The evolution of $A$ is specified by unitary operators $V_\sigma$, $\sigma \in \Gamma$, acting on the complex linear space $\mathbb{C}^Q$. For details, see [Gr99], for example.

We order the states in $Q$, representing the initial vector as $(1, 0, \ldots, 0)$. The evolution is then at each step realized by multiplying a state vector by a transition matrix. There are two types of QFA: those that stop upon reaching a halting state with non-zero amplitude, and those that read the whole input word and stop only upon reading the end marker $\$$. The type affects only the fitness function.

## 2 Genetic operations

Genetic operations may by done directly on the unitary transition matrices defining QFA. The advantage of doing this is easy QFA evaluation, the disadvantage is precision loss while generating QFA and the resulting difficulty to preserve unitarity.

Alternatively, unitary matrices being exponentials of the Hermitian ones, genetic operations may be applied to Hermitian matrices. Simple genetic operations preserving the Hermitian property are possible, hence the automatic preservation of unitarity of automaton transition matrices. However, it may be difficult to evaluate a QFA in terms of its Hermitian matrices.

### 2.1 Operations on unitary matrices

Two kinds of *mutation* operation on unitary matrices $U$ are natural to consider: (i) multiplication of $U$ by a unitary matrix, and, (ii) arbitrary perturbation of $U$ followed by an orthonormalization of its columns.

Operations of the first kind contain in particular the operations of interchanging two rows or two columns of $U$; the complexity of this is $\mathcal{O}(n^3)$. They also contain *rotation* transformations of a state vector of an automaton in one of the plains in its state space $\mathbb{C}^Q$; for random rotation angle generation, exponential scale could be used, $\alpha = \frac{\pi}{2}\beta^M$, where $\beta$ is a random number in the interval $[0; 1]$ and $M$ is positive integer. The complexity of rotation is $\mathcal{O}(n)$.

Operations of the second kind may further consider random order selection in the orthogonalization process. The complexity of these operations is $\mathcal{O}(n^3)$.

One of the main ideas of genetic programming is genetic information exchange between population individuals. This is the *crossing* operation. Here, genetic information is the set of transition matrices of a QFA. As crossing operation, we may then take the exchange of two transition matrices $V_\sigma$ and $V'_\sigma$ of randomly selected automata $A$ and $A'$, for a randomly selected $\sigma \in \Gamma$.

### 2.2 Operations on Hermitian matrices

Every unitary matrix $U$ may be written in the form $U = \exp iH$, where $H$ is a Hermitian (self-adjoint) matrix, $H^\dagger = H$. As *mutation* of $H$ we may take the exchange of any of its elements with the corresponding adjoint element. As *crossing*, we could take the exchange of suitable submatrices of two transition matrices of randomly selected automata for a randomly selected $\sigma \in \Gamma$; However, the complexity of this is at least $\mathcal{O}(n^3)$ and the results are harder to interpret than for unitary operators.

## 3 Fitness function

The fitness function is the most complicated part of the design. It should discriminate between good and bad automata, but not between automata, which are about equally good.

In general, a fitness function cannot test every QFA on all words. It could be reasonable to initially generate a large number of words with end markers, some in the target language and some not, and to test QFA on subsets of these, arbitrarily chosen at every selection stage. A subset of tests could be generated starting with an empty set each time, or changed a little on every new selection stage.

## 4 Selection

There are three basic ways to build the selection function [Kr02]:

*Roulette Wheel.* The probability of a QFA to be thrown out from a population depends on its fitness: the greater the fitness, the greater the probability. Two QFA with equal fitness have equal chances to be thrown out of the population.

*Tournament.* Three QFA are randomly selected and their ratings are compared. The best one goes to the next generation. Two others take tournament again.

*Like Love.* Two random selections are done. The first selects one QFA; let it be $X$. The second selects randomly a set $Y$ of QFA; it could contain all automata but $X$. The automata in $Y$ are compared to $X$, and the first better than $X$ passes to the next generation; if there are no such, $X$ passes.

## 5 Open problems

I only mention the problem of effective representation of QFA matrices (as a set of rotations, for example), and the problem of effective test generation.

## References

[AF98] A. Ambainis and R. Freivalds. *1-way quantum finite automata: strengths, weaknesses and generalizations.* Proc. 39th IEEE Conf. on Foundations of Computer Science, 1998, pp. 332-341.

[BFS00] J. Bārzdiņš, R. Freivalds, and C. H. Smith. *Towards a logic of discovery.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning, Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 110-120.

[BFS01] J. Bārzdiņš, R. Freivalds, and C. H. Smith. *Towards axiomatic basis of inductive inference.* Lecture Notes in Computer Science 2138, 2001, pp. 1-13.

[BFG00] R. Bonner, R. Freivalds and R. Gailis. *Undecidability of 2-tape Quantum Finite Automata.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning, Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 93 - 100.

[Gr99] J. Gruska. *Quantum Computing.* McGraw Hill, 1999.

[KW97] A. Kondacs and J. Watrous. *On the power of quantum finite state automata.* Proc. 38th Symp. on Foundations of Computer Science, 1997, pp. 66-75.

[Kr02] J. Krutjko. *Genetic programming application to solve the problem of "the robot going around the wall".* University of Latvia, Riga, 2002. (in Latvian)

[KK61] G. A. Korn and T. N. Korn. *Mathematical handbook for scientists and engeneers.* McGraw Hill, 1961.

# Inductive Inference of Logic Formulas

Dāvis Kūlis*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia

**Abstract.** This paper continues the research of the synthesis of formulas of many-sorted first order predicate logic with equality, from finite examples. Necessary and sufficient conditions for synthesisability and uniformity are given.

## 1 Introduction

This paper continues the research started in [1], where we considered a UML class diagram synthesis problem ([2, 3]) and generalized it to synthesis of formulas of many-sorted first order logic with equality (further in text simply 'logic') from finite examples. The precise definition of the synthesis problem was based on the notion of language uniformity. It was shown in [1] that not all languages are uniform and that not all uniform languages can be synthesized, but only necessary conditions for uniformity and sufficient conditions for synthesisability were found. Presently, we give necessary and sufficient conditions for uniformity and synthesisability, along with new concepts necessary for proofs.

## 2 Definitions and notation

To make this paper self-contained, we recall some definitions from [1]:

A *signature* $\Sigma$ is a tuple $< C, A >$, where $C$ is a finite set of class names, and $A$ is a finite set of association names with fixed arity and argument types (class names); from now on, let a signature $\Sigma$ be fixed.

A *language* $L$ is a (possibly infinite) set of logic formulas in the given signature containing at least one false formula.

An *example* $s$ is a finite non-empty model in the given signature. Two examples $s_1$ and $s_2$ are considered *equal* iff they can be mapped as annotated graphs preserving class and relation name correspondence, as depicted in Fig. 1.

The *size* $|s|$ of example $s$ is the total number of objects and links in it.

A *sample sequence* $S =< s_1, s_2, \cdots >$ is an infinite subsequence of (possibly repeating) examples from the given signature. For each sample sequence there is a corresponding *sample set* consisting of all examples from the sequence. Though a sample sequence is always infinite, the corresponding sample set can be finite, as the sequence may contain repetitions.

* Research supported by Grant No. 02.0002.1.1 from Latvian Council of Science.

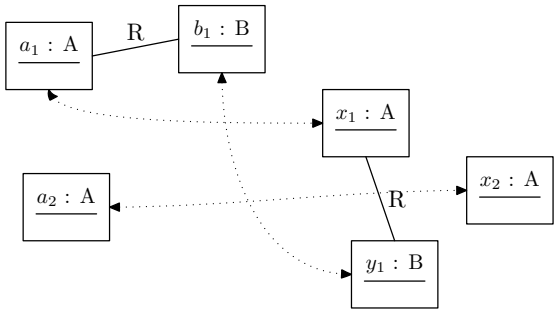


**Fig. 1.** Mapping of examples

The set of all examples on which the formula $f$ is true is denoted by $\mathcal{E}_f$. Formulas $f_1$ and $f_2$ are *equal* iff $\mathcal{E}_{f_1} = \mathcal{E}_{f_2}$. $\mathcal{E}_f$ lets us depict formula $f$ in diagram by a sample set. However, for clarity, instead of writing $\mathcal{E}_f$, we will write $f$.

A formula $f$ *covers* an example $s$ ($f \rightarrow s$) iff $f$ is true on $s$, i.e. when $s \in \mathcal{E}_f$. The fact that a formula $f$ does not cover an example $s$ is denoted by $f \nrightarrow s$. The coverage concept can be extended to sets of examples and formulas. We thus say say that a formula $f$ covers a sample set $S$ ($f \rightarrow S$) iff $f$ is true on every sample $s \in S$ (i.e. $S \subseteq \mathcal{E}_f$), and, a set of formulas $F$ covers a sample set $S$ ($F \rightarrow S$) iff every formula $f \in F$ is true on every sample $s \in S$.

## 3 Strongest formula

There exists an obvious solution for the synthesis problem – the identically true formula $\mathcal{T}$. To obtain nontrivial results we must constrain the result of the synthesis algorithm. As such constraint we choose the minimization of the example set covered by resulting formula. To make this idea precise, we introduce the notion of the strongest formula.

**Definition 1.** *Formula $f_1$ is* stronger *than formula $f_2$ ($f_1 < f_2$) iff $\mathcal{E}_{f_1} \subset \mathcal{E}_{f_2}$ as shown on Fig. 2.*

Further, we write $f_1 \leq f_2$ when $f_1$ is stronger than or equal to $f_2$, and, $f_1 \div f_2$ when $f_1$ is not comparable with (neither equal to, nor stronger or weaker than) $f_2$, as in Fig. 3.

**Definition 2.** *A formula $f$ from language $L$ is* strongest *for sample set $S$ iff it covers $S$ and $L$ does not contain any other formula $f'$ stronger than $f$ covering $S$.*

There can be more than one strongest formula for a given language and sample set, as illustrated by Fig. 4; let $M_S^L$ denote the set of all strongest formulas for sample set $S$ and language $L$.

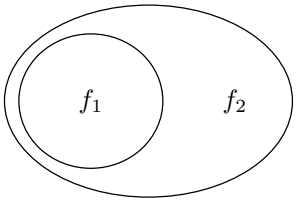


**Fig. 2.** Formula $f_1$ is stronger than $f_2$ ($f_1 < f_2$)

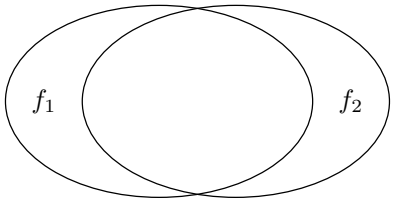


**Fig. 3.** Incomparable formulas $f_1 \div f_2$

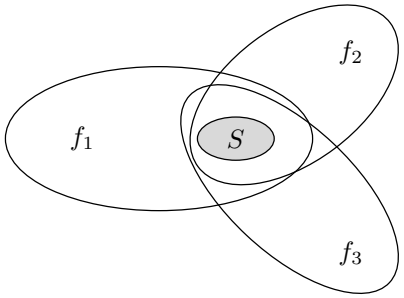


**Fig. 4.** Strongest formulas

## 4 Uniformity

**Definition 3.** *A language $L$ is* uniform *iff for every sample set $S$ it has exactly one strongest formula, i.e.* $\forall S|M_S^L| = 1$.

Given a uniform language $L$, denote by $\mu_S^L$ the single strongest formula for the sample set $S$; in other words, $M_S^L = \{\mu_S^L\}$, then.

It was shown in [1] that there exists both uniform and nonuniform languages. We also proved some necessary conditions for uniformity. In this paper, we prove necessary and sufficient conditions for uniformity. To do this, we introduce several additional concepts.

**Definition 4.** *A sample set $K$ is an* intersection *of (probably infinite) set of formulas $F = \{f_1, f_2, \dots\}$ iff:*

$$\forall f \in F(f \to K), \tag{1}$$

*and,*

$$\forall x \notin K \exists f \in F(f \nrightarrow x) \ . \tag{2}$$

In other words, $K = \prod_{f \in F} \mathcal{E}_f$.

**Definition 5.** *A formula $f'$ is an* intersection *of the set set of formulas $F = \{f_1, f_2, \dots\}$ iff the sample set $\mathcal{E}_{f'}$ is an intersection of $F$.*

It may be noted that this definition allows for a set of formulas to intersect on a false formula (which covers empty set). This is the reason why, by convention, the false formula is an element of every language.

**Theorem 1.** *A language $L$ is uniform iff it contains true formula $\mathcal{T}$, and, the intersection of any subset of formulas $F \subseteq L$ is also a formula in $L$.*

*Proof.* First we prove that these are *necessary* conditions, i.e. if any of them do not hold, then $L$ is not uniform.

If $L$ does not contain $\mathcal{T}$, then it cannot cover sample set containing all examples from the given signature. So in this case $L$ cannot be uniform.

Next, assume that $L$ is uniform, and contains a subset of formulas $F \subseteq L$ such that $F$ intersects on sample set $K$, but not on any formula from $L$ (i.e. $\forall f \in L(\mathcal{E}_f \neq K)$). However, as $L$ is uniform, $K$ has the strongest formula $\mu_K^L$. Since $\mu_K^L$ covers $K$ and no formula $f \in L$ has $\mathcal{E}_f = K$, then $K$ is proper subset of $\mathcal{E}_{\mu_K^L}$. Therefore there exists an example $s$ such that

$$s \in \mathcal{E}_{\mu_K^L} \text{ and } s \notin K \ . \tag{3}$$

Figure 5 illustrates this construction.

Now applying (2) to the sample $s$ we get that

$$\exists f \in F(f \nrightarrow s) \ . \tag{4}$$

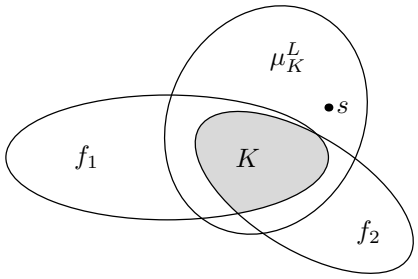


**Fig. 5.** Nonuniform intersection

From (3), (4) we see that $\mu_K^L \not\leq f$. Thus there exists some formula $f$ which by (1) covers $K$ and is stronger than or incomparable to the strongest formula for $K$. This is a contradiction, and our assumption of uniformity of $L$ is wrong. The necessity of both uniformity conditions is thus proved.

Next, we prove that these are *sufficient* conditions, i.e. that if both uniformity conditions hold for a language, then it is uniform.

Assume the contrary, that both uniformity conditions hold for $L$, but it is not uniform. This means there exists a sample set $S$ having several or no strongest formulas.

First, consider a case when $S$ has several (mutually incomparable) strongest formulas $F = \{f_1, f_2, \dots\}$. According to uniformity conditions $F$ intersects on some formula $f' \in L$, as depicted on Fig. 6.

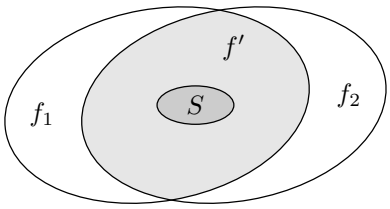


**Fig. 6.** Intersection of strongest formulas

$S \subseteq \mathcal{E}_{f'}$, otherwise $\exists s \in S$ such that $s \notin \mathcal{E}_{f'}$, and by (2) $\exists f \in F(f \nrightarrow s)$, from what follows $f \nrightarrow S$, which is a contradiction as $f \rightarrow S$ because it is the strongest formula for $S$. Thus

$$f' \rightarrow S \ . \tag{5}$$

It can also be seen that $\forall f \in F(\mathcal{E}_{f'} \subseteq \mathcal{E}_f)$ (since $f \rightarrow \mathcal{E}_{f'}$ by (1)) and no $f \in F$ equals $f'$ (or this $f$ would be stronger than other formulas from $F$, but they are incomparable). Therefore $\forall f \in F(\mathcal{E}_{f'} \subset \mathcal{E}_f)$ and

$$\forall f \in F(f' < f) \ . \tag{6}$$

Together (5) and (6) show that there exists formula $f'$ covering $S$ that is stronger than any of the strongest formulas for $S$. This is a contradiction, so such set $F$ cannot exist.

Finally consider a case when $S$ does not have any strongest formula. This is possible either when no formula from $L$ covers $S$ (not possible as $L$ contains $\mathcal{T}$), or for any formula $f \to S$ exists a stronger formula $g \in L$ such that $g < f$ and $g \to S$.

Assume that the latter case is possible, and construct a bset $F$ of all formulas from $L$ covering $S$. $F$ intersects on some formula $f' \in L$. Similarly as shown above, $f' \to S$.

According to our assumption there should exist some $g \in L$ such that $g < f'$ and $g \to S$. From $g < f'$ follows that

$$\mathcal{E}_g \subset \mathcal{E}_{f'} \ . \tag{7}$$

On the other hand $g \in F$ as $g \to S$. Therefore according to (1) $g \to \mathcal{E}_{f'}$ and

$$\mathcal{E}_{f'} \subseteq \mathcal{E}_g \ . \tag{8}$$

It can be seen that (7) and (8) contradict each other and our assumption is wrong. Thus sufficiency of uniformity conditions is also proved. □

## 5 Synthesis

**Definition 6.** *An algorithm $\mathcal{A}_L$ is a* synthesis algorithm *for a uniform language $L$ iff for every sample sequence $S =< s_1, s_2, s_3, \cdots >$ the following holds. For each initial fragment $S_i =< s_1, s_2, s_3, \ldots, s_i >$ the algorithm $\mathcal{A}_L$ computes some formula $f_i \in L$ such that $f_i \to S_i$; there exists natural $n$ such that $\forall i > n(f_i = f_n)$, and, $f_n = \mu_S^L$.*

**Definition 7.** *A language $L$ is* synthesisable *iff there exists a synthesis algorithm $\mathcal{A}_L$ for $L$.*

In [1] it was shown that there are some unsynthesisable languages, but only the sufficient conditions for synthesisability were given. Presently, we give necessary and sufficient conditions for synthesisability of uniform languages.

**Theorem 2.** *A computably enumerable uniform language $L$ is synthesisable iff it does not contain an infinite subsequence of weakening formulas $F$*

$$f_1 < f_2 < f_3 < \ldots \ . \tag{9}$$

To prove the sufficiency of this condition we need the following lemmas:

**Lemma 1.** *Given a uniform language $L$, the strongest formulas for two sample sets $S_1$ and $S_2$ such that $S_1 \subset S_2$ have the following property:*

$$\mu^L_{S_1} \leq \mu^L_{S_2} \ . \tag{10}$$

*Proof.* Assume the contrary – there exists two such sample sets $S_1$ and $S_2$ such that $S_1 \subset S_2$ for whom

$$\mu^L_{S_1} > \mu^L_{S_2} \tag{11}$$

or

$$\mu^2_{S_1} \div \mu^L_{S_2} \ . \tag{12}$$

By definition $\mu^L_{S_2} \rightarrow S_1$, so (11) is not true.

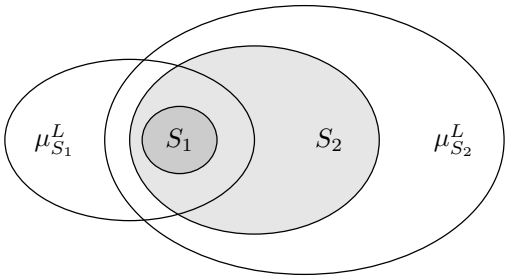


**Fig. 7.** Case $\mu^L_{S_1} \div \mu^L_{S_2}$

Formula (12) is also false, otherwise, due to the uniformity condition, we could find another formula $f$ covering $S_1$ stronger than $\mu^L_{S_1}$, which is impossible as $\mu^L_{S_1}$ already is the strongest formula.

So our assumption is incorrect and the lemma is proved. □

**Lemma 2.** *Given a uniform language $L$ and two sample sets $S_1 \subset S_2$, if there exists formula $f \in L$ such that $f \rightarrow S_1$ and $f \nrightarrow S_2$ then $\mu^L_{S_1}$ is strictly stronger than $\mu^L_{S_2}$:*

$$\mu^L_{S_1} < \mu^L_{S_2} \ . \tag{13}$$

*Proof.* Assume the contrary – there exists two such sample sets $S_1 \subset S_2$ and formula $f \in L$ such that $f \rightarrow S_1$ and $f \nrightarrow S_2$, but

$$\mu^L_{S_1} \not< \mu^L_{S_2} \ . \tag{14}$$

According to Lemma 1 $\mu^L_{S_1} \leq \mu^L_{S_2}$, so together with (14) we get

$$\mu^L_{S_1} = \mu^L_{S_2} \ . \tag{15}$$

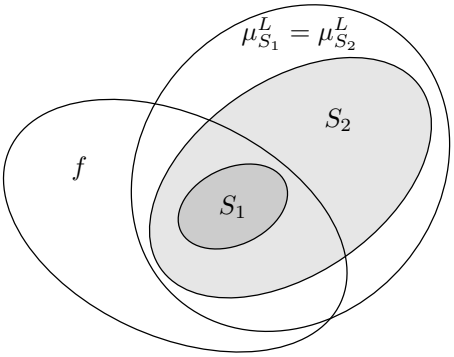


**Fig. 8.** Formula $f \to S_1$ and $f \nrightarrow S_2$

This is illustrated in Fig. 8.

As $f \to S$, then by definition $\mu^L_{S_1} \leq f$, which we can now substitute to $\mu^L_{S_2} \leq f$ due to (15). Also by definition $\mu^L_{S_2} \to S_2$, which combined with $\mu^L_{S_2} \leq f$ leads to $f \to S_2$. But this is a contradiction, so the lemma is proved. □

*Proof of the Theorem 2.* First we prove *necessity* of the synthesisability condition – if $L$ contains subsequence (9) then it cannot be synthesisable.

We assume $L$ is synthesisable but contains a sequence of formulas $f_1 < f_2 < f_3 < \ldots$, and then construct a sample sequence $S$ on which any possible synthesis algorithm $\mathcal{A}_L$ breaks down.

We will construct $S$ starting with one example and then adding new examples to the initial fragment of $S$. Here and further in proof such initial fragment of $S$ with $i$ elements we denote by $S_i$. Along with the construction of $S_i$ we will maintain index $k(i)$ with the following property: $S_i$ is covered by $f_{k(i)}$, but is not covered by any of $f_{k(i)-1}, f_{k(i)-2}, \ldots$ (i.e. $\forall x > 0(f_{k(i)-x} \nrightarrow S_i)$). This is illustrated by Fig. 9.

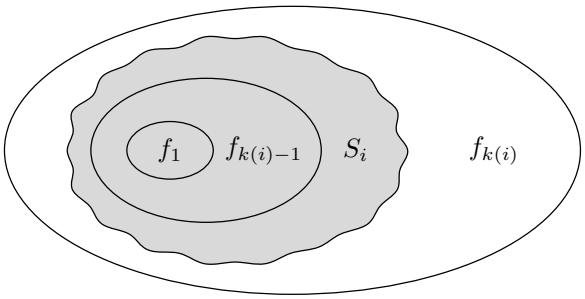


**Fig. 9.** Relation between $S_i$ and $k(i)$

Initially we choose $k(i) = 1$ and $S_1 = \{s_1\}$ such that $s_1 \in \mathcal{E}_{f_1}$. Then we apply following recursive procedure to get $S_{i+1}$ from $S_i$ – feed $S_i$ into $\mathcal{A}_L$ and depending on the result $f$:

1. if $f < f_{k(i)}$ then choose the next example $s_{i+1}$ from the set $\mathcal{E}_{f_{k(i)}} \setminus \mathcal{E}_f$ and $k(i+1) = k(i)$;
2. if $f = f_{k(i)}$ then choose the next example $s_{i+1}$ from the set $\mathcal{E}_{f_{k(i)+1}} \setminus \mathcal{E}_{f_{k(i)}}$ and $k(i+1) = k(i) + 1$, finally
3. if $f > f_{k(i)}$ then $s_{i+1} = s_i$ and $k(i+1) = k(i)$.

So any synthesis algorithm will either never converge on any result (due to option 1 and 2), or it will converge on some formula which is not the strongest for $S$ (option 3). Therefore such algorithm cannot be synthesis algorithm for $L$, and $L$ is not synthesisable.

Next we prove *sufficiency* of the synthesisability condition – if $L$ does not contain the subsequence (9) then $\mathcal{A}_L$ exists and $L$ is synthesisable. For the rest of the proof we assume that $L$ has been enumerated as $f_1, f_2, \ldots, f_i, \ldots$.

Consider an algorithm $\mathcal{A}_L$, which performs the following steps on the input sequence $S_i =< s_1, s_2, \ldots, s_i >$:

1. Finds the index $l_i$ of first formula $f_{l_i}$ such that $f_{l_i} \rightarrow S_i$. This step will finish, as every uniform language contains at least $\mathcal{T}$.
2. Computes the set $P_i \subseteq L$ containing all formulas $f_1, f_2, \ldots, f_{\max(l_i,i)}$. If $L$ is finite and has less than $i$ elements, then $P_i = L$.
3. Computes the set $R_i \subseteq P_i$ such that $R_i \rightarrow S_i$. $R_i$ will not be empty as it will contain at least $f_{l_i}$.
4. For each $f \in R_i$ computes the sample set $E_i(f) \subseteq \mathcal{E}_f$ which contain all examples covering $f$ of size $i$ and smaller.
5. Finds the set of the locally strongest formulas $M_i \subseteq R_i$ such that

$$\forall f \in M_i \nexists f' \in R_i (E_i(f) \supset E_i(f')) \ . \tag{16}$$

   $M_i$ will not be empty, as $R_i$ is not empty.
6. Return $r_i \in M_i$ with the minimal index according to enumeration of $L$.

To prove that this is a synthesis algorithm for $L$ we assume that $S$ is fixed, and consider two mutually exclusive cases which characterize the way examples are ordered in $S$:

– "significant" examples are scattered throughout $S$, and after any $n$ examples we can find some example showing one of our hypotheses (formulas in $R_i$) is wrong. This is equivalent to the statement that for every $n$ there will be some $R_i$ $(i > n)$ that cannot cover sample set $S_j$ (obviously $i < j$):

$$\forall n \exists i > n \exists j (R_i \nrightarrow S_j) \ ; \tag{17}$$

– after certain number $n$ the sample subsequence $S_n$ contains all "significant" examples, and for every sequence $S_i$ longer than $n$ $R_i$ will cover the whole $S$ (or any subsequence $S_j$ of it, which is the same):

$$\exists n \forall i > n \forall j (R_i \to S_j) \ . \tag{18}$$

First we can show that (17) is false. Choose arbitrary $n_1$. According to (17) there exists corresponding $i_1$, $R_{i_1}$, $j_1$ and $S_{j_1}$. Thus we can build infinite sequence, each time choosing $j_k$ for the next $n_{k+1}$.

As $R_{i_k}$ does not cover $S_{j_k}$, then there is some formula $f \in R_{i_k}$ such that $f \nrightarrow S_{j_k}$. But according to construction of $A_L$ $R_{i_k}$ covers $S_{i_k}$ and any subset of it, so $f \to S_{i_k}$ and $S_{i_k}$ must be a proper subset of $S_{j_k}$. Hence according to Lemma 2 $\mu^L_{S_{i_k}} < \mu^L_{S_{j_k}}$. On the other hand, as $j_k = n_{k+1} < i_{k+1}$ then $S_{j_k} \subset S_{i_{k+1}}$ and according to Lemma 1 $\mu^L_{S_{j_k}} \leq \mu^L_{S_{i_{k+1}}}$. Combining those equations we can build an infinite sequence:

$$\mu^L_{S_{i_1}} < \mu^L_{S_{j_1}} \leq \mu^L_{S_{i_2}} < \mu^L_{S_{j_2}} \leq \mu^L_{S_{i_3}} < \dots \tag{19}$$

which contains a subsequence

$$\mu^L_{S_{i_1}} < \mu^L_{S_{i_2}} < \mu^L_{S_{i_3}} < \dots \ . \tag{20}$$

But this contradicts with synthesisability condition, so (17) is false.

Therefore (18) is true. As stated above, it is equivalent to

$$\exists n \forall i > n (R_i \to S) \ . \tag{21}$$

So after some sample set $S_n$ output of $\mathcal{A}_L$ will cover the whole $S$. Now we have to prove that this output will eventually stabilize on strongest formula of $L$.

By $k$ we denote the index of $\mu^L_S$ in the enumeration of $L$ (i.e. $\mu^L_S = f_k$). Since $\mu^L_S$ covers any subset of $S$, then according to construction of $\mathcal{A}_L$ for each $i > k (\mu^L_S \in R_i)$. So for any input with size $\max(n, k)$ or longer the set $R_i$ will contain $\mu^L_S$, and $\mu^L_S$ will be stronger than any other fromula from $R_i$ as all formulas from $R_i$ cover $S$. Last thing we have to prove is that we can separate $\mu^L_S$ from $R_i$.

By definition of the strongest formula

$$\forall f_i \in L (f_i \to S \Rightarrow \mathcal{E}_{\mu^L_S} < \mathcal{E}_{f_i}) \ , \tag{22}$$

and for every $f_i \to S$ there exists an example $p_i$ such that $f_i$ covers $p_i$, but $\mu^L_S$ does not. Thus, if on step 4 of $\mathcal{A}_L$ we construct sets $E_x(\mu^L_S)$ and $E_x(f_i)$ large enough to include $p_i$, we will see that $f_i$ is weaker.

In fact, for $\mathcal{A}_L$ to return $\mu^L_S$ all we have to check is that all formulas prior to $\mu^L_S$ in enumeration (i.e. with indexes less than $k$) and covering $S$ are weaker than $\mu^L_S$. Therefore we introduce $m = \max_{i<k} |p_i|$. Obviously

$$\forall i < k \forall j > m (f_i \to S \Rightarrow E_j(\mu^L_S) \subset E_j(f_i)) \ . \tag{23}$$

So for any input with size $\max(n, k, m)$ or longer $M_i$ will contain no formula with index less than $k$, either because it does not cover $S$, or because it is weaker than $\mu_S^L$. Thus $\mu_S^L$ will be first element in $M_i$ and $\mathcal{A}_L$ will return it.

So the sufficiency of the synthesisability condition is also proved. □

## 6 Conclusions

In this paper we have defined and explored notions of the strongest formula, language uniformity, and, synthesisability. We have proved necessary and sufficient conditions for uniformity and synthesisability.

In summary, a language is synthesisable if it is computably enumerable, it contains the true formula, the intersection of any subset of language formulas is also a formula, and, it contains no infinite sequence of weakening formulas. These conditions hold for quite a broad class of languages, including such important languages as UML class diagrams with four multiplicity constraints 1, 0..1, 1..∗, 0..∗.

We finally note that these results for logic formulas can be easily generalized for recursive predicates.

## References

1. J. Bārzdiņš, Ģ. Linde and D. Kūlis. *Learning of formulae from finite examples.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning, Proc. Int. Workshop, Riga, 1999, pp. 200-208.
2. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorsen. *Object-oriented modeling and design.* Prentice Hall, 1991.
3. OMG Unified Modeling Language Specification v1.3, June 1999, `http://www.rational.com/uml`

# Quantum Learning by Finite Automata

Richard Bonner[1] and Rūsiņš Freivalds[2]

[1] Department of Mathematics and Physics
Mälardalens University, Sweden
richard.bonner@mdh.se
[2] Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia*
Rusins.Freivalds@mii.lu.lv

**Abstract.** Freivalds and Smith [FS92] proved that probabilistic limited memory inductive inference machines can learn some classes of total recursive functions with probability 1, which cannot be learned by deterministic limited memory inductive inference machines. We introduce quantum limited memory inductive inference machines as quantum finite automata used as inductive inference machines. Our main result shows that quantum limited memory inductive inference machines can learn classes of total recursive functions not learnable by any deterministic, and not even by probabilistic, limited memory inductive inference machines.

## 1 Introduction

E. M. Gold, in a seminal paper [Gold67], defined the notion of *identification in the limit*. This definition concerned learning by algorithmic devices now called *inductive inference machines* (IIMs). An IIM inputs the graph of a total recursive function, an ordered pair at a time, and while doing so, outputs computer programs. Since we will only discuss the inference of total recursive functions, we may assume, without loss of generality, that the input is received by an IIM in its natural domain increasing order, $f(0)$, $f(1)$, $\cdots$. An IIM, on input from a function $f$ will output a potentially infinite sequence of programs $p_0$, $p_1$, $\cdots$. The IIM *converges* if either the sequence is finite, say of length $n+1$, or there is a program $p$ such that $p_i = p$ for all but finitely many $i$. In the former case, we say that the IIM converges to $p_n$, and in the latter case, to $p$. In general, there is no effective way to tell when, and if, an IIM has converged.

Following Gold, we say that an IIM $M$ *identifies* a function $f$ in the limit (written: $f \in EX(M)$), if, when $M$ is given the graph of $f$ as input, it converges to a program $p$ that computes $f$. If an IIM identifies some function $f$, then some form of learning must have taken place, since, by the properties of convergence, only finitely much of the graph of $f$ was known by the IIM at the (unknown)

* Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission, and the Swedish Institute, Project ML-2000.

point of convergence. The terms *infer* and *learn* will be used as synonyms for *identify*. Each IIM will learn some set of recursive functions. The collection of all such sets, over the universe of effective algorithms viewed as IIMs, serves as a characterization of the learning power inherent in the Gold model. This collection is symbolically denoted by $EX$ (for *explanation*) and is rigorously denoted by $EX = \{U|\exists M(U \subseteq EX(M))\}$. Similarly, we say that an IIM $M$ *identifies* a function $f$ finitely (written: $f \in FIN(M)$), if, when $M$ is given the graph of $f$ as input, it outputs exactly one program $p$ that computes $f$, and then the machine stops. $FIN = \{U|\exists M(U \subseteq FIN(M))\}$.

These collections are set-theoretically compared with the collections that arise from other models, which we discuss below. Many intuitions about machine learning have been gained by working with Gold's model and its derivatives. For a more detailed explanation of this influence we refer to the paper by Arikawa and Mukouchi [AM95].

In the next section we describe the variants of Gold's model examined in this paper.

## 2 Limited memory learning

A study of inference machines with limited memory (the current guess and the next one, or selected data only) was initiated by Wiehagen [Wieh76] and pursued by Arikawa and his students [AH87,AN91], by Wiehagen and Zeugmann [WZ94], and others. The conclusion reached in this research was that restricting the data available to the inference machine also reduces its learning potential.

We use models as close as possible to the one in the conference paper [FS92] by Freivalds and Smith, later incorporated into a larger journal paper [FKS95].

To insure an accurate accounting of the memory used by an IIM, we will henceforth assume that each IIM receives its input in such a way that it is impossible to back up and reread some input after another has been read. To circumvent the use of coding techniques, the memory used will be measured in bits, as opposed to integers. Under these conventions, we say that a set $U \subseteq LEX(M)$ iff there is a constant $c$ such that for any $f \in U$, $M$ uses no more than $c$ bits of memory, exclusive of the input, and $f \in EX(M)$. One formalization of this notion considers the memory limited IIMs as Turing machines with input tape and work tape. The input tape is read only once (one way) and the work tape has only $c$ bits of storage capacity. An equivalent formalization is to view memory limited IIMs as finite automata. The collection of all sets of functions inferable by limited memory inference machines is denoted by $LEX$, where $LEX = \{U|\exists M(U \subseteq LEX(M)\}$.

A few more technical definitions are needed. Natural numbers ($\mathbf{N}$) will serve as names for programs. The function computed by program $i$ will be denoted by $\phi_i$. It is assumed that $\phi_0$, $\phi_1$, $\cdots$, forms an acceptable programming system [Rog87]. Sometimes it will be convenient to represent a total function by a sequence of values from its graph. Such a representation is called a *string* representation. So, for example, the sequence $01^20^43^\infty$ represents the (total) function

equal to zero if $x = 0$ or $3 \leq x \leq 6$, equal to one if $1 \leq x \leq 2$, and equal to three otherwise. The function in this example has two blocks of consecutive 0's, one of length 1 and the other of length 4.

In order to get a rough idea of the relative learning power of $LEX$ type inference, we will employ the set of functions of finite support and the set of self describing functions. These sets were introduced in [BB75] and used in [Fre91,FKS95a] to separate various classes of learnable sets of functions. Let $U_0 = \{f | f \text{ is recursive and } \forall^\infty x(f(x) = 0))\}$ and $U_1 = \{f | f \text{ is recursive and } \phi_{f(0)} = f\}$.

The following propositions were proved in [FS92]:

**Proposition 1.** $U_1 \in LEX$.

**Proposition 2.** $U_0 \notin LEX$.

**Proposition 3.** $LEX \subset EX$.

Probabilistic inductive inference machines were introduced in [Pitt89] and studied further in [PS88]. A probabilistic inductive inference machine is an IIM that makes use of a fair coin. Formally speaking, one needs to introduce identifiability with a specific coin outputting 0 with probability $\alpha$, and outputting 1 with probability $1 - \alpha$, and to show that identifiability with any specific $\alpha$ is no less powerful than identifiability with $\alpha = \frac{1}{2}$. Hence, in our notation, we denote only the probability of the correct result but not the $\alpha$ in the coin. We say that $f \in PrEX(M)$ if $M$ learns $f$ with probability $p$, $0 \leq p \leq 1$. The collection $PrEX\langle p \rangle$ is defined to be $\{U | \exists M(U \subseteq PrEX(M)\langle p \rangle)\}$. Pitt [Pitt89] showed that for $p > \frac{1}{2}$, $PrEX\langle p \rangle = EX$. Limiting the memory available to a probabilistic IIM, according to our conventions, gives rise to the class $PrLEX\langle p \rangle$.

Freivalds and Smith [FS92] proved that probabilistic limited memory machines can learn with probability 1 a class which cannot be learned by any deterministic limited memory machine:

**Theorem 1.** [FS92] *There is a class $U$ of total recursive functions such that $U \in PrLEX\langle 1 \rangle \setminus LEX$.*

This is somewhat surprising since 'with probability 1' is often considered as an equivalent to 'deterministic' (which it is not, of course).

It seems that a more natural definition of limited memory inductive inference machines would involve the notion of finite automata introduced by Blum, Shub and Smale [BSS89]. The BSS-automata can process arbitrarily large integers but they cannot distinguish large numbers. These automata can store integers in a finite number of registers and move the integers from one register to another. In our case (when these automata are used as inductive inference machines), the new definition gives additional possibility to output target function values stored at earlier moments. However this modified definition would again produce the same result as Theorem 1. This is a direct consequence of the fact that the class of functions considered in Theorem 1 deterministically cannot be learned even by the modified limited memory inductive inference machines.

Our main results relate to deterministic (LFIN) and probabilistic (PrLFIN) limited memory finite learning of recursive functions. This means that the result is to be produced after a finite number of steps, and the learning process terminates after this output(finite learning) and the learning is done by finite memory inductive inference machines. Strangely enough, we have found no published results on finite learning by limited memory inductive inference machines.

## 3 Quantum finite automata

Quantum finite automata were introduced by Kondacs and Watrous [KW97] (another but much weaker definition was considered by Moore and Crutchfield [MC00], the technical report version of which was published the same year). Informally, quantum automata are very similar to probabilistic automata, but use unitary instead of stochastic transition matrices. Formally, 1-way quantum finite automaton (QFA) is a tuple

$$M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$$

where $Q$ is a finite set of states, $\Sigma$ is an input alphabet, $\delta$ is a transition function, $q_0 \in Q$ is a starting state and $Q_{acc} \subset Q$ and $Q_{rej} \subset Q$ are sets of accepting and rejecting states. The states in $Q_{acc}$ and $Q_{rej}$ are called *halting states* and the states in $Q_{non} = Q - (Q_{acc} \cup Q_{rej})$ are called *non-halting states*. # and \$ are symbols that do not belong to $\Sigma$. We use # and \$ as left and right endmarker, respectively. The *working alphabet* of $M$ is $\Gamma = \Sigma \cup \{\#, \$\}$.

The work of a probabilistic one-way finite automaton can be described by the distribution of probabilities of all the internal states of the automaton at the current moment. This distribution can be imagined as a row-vector $(\xi_i)$ of probabilities $\xi_i$ to be in the state $q_i$, $1 \leq i \leq s = |Q|$. When the next input symbol is read from the input, this row-vector is multiplied to a stochastic matrix $(\eta_{ij})$, $1 \leq i, j \leq s$. Finally, when all the input word has been read from the input, the probabilities of all the accepting states are totalled. We say that the word is accepted, if the total of these probabilities exceeds $\frac{1}{2}$.

Similarly, the work of a quantum one-way finite automaton can be described by the distribution of amplitudes (being complex numbers) of all the internal states of the automaton at the current moment. This distribution can be imagined as a row-vector $(\xi_i)$ of complex numbers, $|\xi_i|^2$ being the probability to be in the state $q_i$, $1 \leq i \leq s = |Q|$. When the next input symbol is read from the input, this row-vector is multiplied to a unitary matrix $(\eta_{ij})$, $1 \leq i, j \leq s$. This gives us the distribution of amplitudes of all the internal states at the next moment. However, in contrast with probabilistic automata, the transformation of the distribution of amplitudes consists of two steps. First, the old row-vector is multiplied to the unitary matrix corresponding to the symbol read from the input. Second, all the halting states are *measured*, i.e. the new amplitude $\alpha$ is replaced by 0 but the value $|\alpha|^2$ is added to the total probability of acceptance (if the halting state is an accepting one) or it is added to the total probability of rejection (if the halting state is a rejecting one).

When an input word $x$ is processed by a quantum finite automaton, we total the probabilities to accept this word at all moments (including that when the end-marker \$ is read from the input). We say that the word is accepted, if the total of these probabilities exceeds $\frac{1}{2}$. Just as in the case of probabilistic finite automata, most of the research on quantum automata concerns language recognition with probability $p$ strictly exceeding $\frac{1}{2}$.

Kondacs and Watrous [KW97] proved that quantum finite automata recognize fewer languages than deterministic finite automata.

**Theorem 2.** ([KW97]) *Every language recognized by 1-way QFA with probability $p > \frac{1}{2}$ is recognizable by some deterministic FA as well. The converse does not hold: the language $\{0,1\}^*1$ recognized by a deterministic FA but not by any 1-way QFA with probability $p > \frac{1}{2}$.*

On the other hand, Ambainis and Freivalds [AF98] proved that the size of quantum finite automata can be exponentially smaller than the size of any equivalent deterministic or even probabilistic finite automata recognizing the same language.

## 4 Quantum learning

Our goal is to consider Gold type identification in the limit by limited memory quantum inductive inference machines. However, it is difficult to find a model of learning, for which quantum learning would give advantages over classical (deterministic or probabilistic) learning. Indeed, it is an easy observation that all functions computable by quantum computers are recursive, and hence no advantages of quantum learning can be proved if unrestricted calculations are allowed. This is why we considered limited memory learning.

By Theorem 2, quantum finite automata are strictly less powerful than deterministic finite automata. Does this mean that quantum limited memory learning is also less powerful than the deterministic one? No.

Our concept of learning is very much limited. Since our inductive inference machine is only a finite automaton but the values of the target function $f$ can be arbitrarily large integers, our machine can only distinguish among the values 0, 1, 2, and, 'a larger integer'. The machine can output only the current value of the target function. (In a more general case, using BSS-type finite automata [BSS89], it would be possible to output an input value previously stored in one of the registers of the IIM.) However the model turns out to be powerful enough to show the advantage of quantum learning over deterministic and probabilistic learning.

We say that $f \in QLFIN(M)$ if the quantum limited memory inductive inference machine $M$ learns $f$ with probability $p$, $0 \leq p \leq 1$. The collection $QLFIN\langle p\rangle$ is defined to be $\{U|\exists M(U \subseteq QLFIN(M)\langle p\rangle)\}$.

Our main result below shows that quantum limited memory inductive inference machines can learn classes of total recursive functions not learnable by any deterministic limited memory inductive inference machines.

## 5 Results

For a numerical function $f$, introduce the working notation $u = |f^{-1}(0)|$ and $v = |f^{-1}(1)|$, i.e. $u$ is the number of values $x$ where $f(x) = 0$, and $v$ is the number of values $x$ where $f(x) = 1$.

**Definition 1.** *Let $V_\epsilon$ be the class of all total recursive functions $f$ such that: (i) $f(0) = 2$, (ii) for some $z = z_f$, $f(y) < 2$ for all $y < z$, $f(y) > 2$ for $y = z$ and for $y = z + 1$, and, $f(y) = 2$ for all $y > z + 1$, and, (iii) either $\cos^2 u < \epsilon$ and $\cos^2 v > 1 - \epsilon$, and $f(z)$ is a program to compute the function $f$, or, $\cos^2 v < \epsilon$ and $\cos^2 u > 1 - \epsilon$, and $f(z + 1)$ is a program to compute the function $f$.*

Let $\langle m, n\rangle$ be the standard pairing function for one-to-one correspondence between pairs of natural numbers and natural numbers, see [Rog87].

**Theorem 3.** (Smullyan [Rog87]) *For arbitrary total recursive functions $g$ and $h$, there exist $m$ and $n$ such that $\phi_m = \phi_{g(\langle m,n\rangle)}$ and $\phi_n = \phi_{h(\langle m,n\rangle)}$.*

This well-known double recursion theorem of Smullyan will be used in the proof of the next theorem.

**Theorem 4.** *The class $V_\epsilon$ is not in LFIN.*

*Proof.* Assume to the contrary that $V_\epsilon \in LFIN(M)$. Let $w$ be the number of the states of the finite automaton serving as our inductive inference machine, and put $c = w!$. Hence, if the automaton receives a sequence of zeros (or a sequence of ones) as part of the input values of the target function $f$, the automaton repeats its internal states with period not exceeding $w$. The automaton is thus not able to notice the presence or absence of any subsequence of zeros, the length of which is a multiple of $c$.

It is well-known (Theorem 6.3 in [Niv67]) that for $\xi$ irrational, the sequence $\xi$, $2\xi$, $3\xi$, $\cdots$, is uniformly distributed modulo 1. Since $\pi$ is well-known to be irrational (e.g. Cor. 2.6 in [Niv67]), it follows that the sequences $\cos^2 m$ and $\sin^2 m$, $m = 1, 2, \ldots$, are dense in the interval $[0, 1]$. Hence, for arbitrary $n_0$ there is $k > 1$ such that $\cos^2 n < \epsilon$ and $\sin^2 n > 1 - \epsilon$, with $n = n(n_0, c) = n_0 + kc$. Similarly, for arbitrary $m_0$ there is $k > 1$ and such that $\cos^2 m > 1 - \epsilon$ and $\sin^2 m < \epsilon$, with $m = m(m_0, c) = m_0 + kc$. Note that $n(n_0, c) > n_0$ and $m(m_0, c) > m_0$.

Now choose $n_0$ and $m_0$ so that $\cos^2 n_0 > 1 - \epsilon$ and $\cos^2 m_0 < \epsilon$. Using the double recursion theorem (Theorem 3) we construct a pair of total recursive functions $f(x)$ and $g(x)$ such that

$$f(x) = \phi_d(x) = \begin{cases} 2 & \text{if } x = 0 \\ 0 & \text{if } 1 \le x \le n_0 \\ 1 & \text{if } n_0 + 1 \le x \le n_0 + m_0 \\ d & \text{if } x = n_0 + m_0 + 1 \\ e & \text{if } x = n_0 + m_0 + 2 \\ 2 & \text{if } x > n_0 + m_0 + 2 \end{cases}$$

and

$$g(x) = \phi_e(x) = \begin{cases} 2 & \text{if } x = 0 \\ 0 & \text{if } 1 \leq x \leq n(n_0, \epsilon) \\ 1 & \text{if } n(n_0, \epsilon) + 1 \leq x \leq n(n_0, \epsilon) + m(m_0, \epsilon) \\ d & \text{if } x = n(n_0, \epsilon) + m(m_0, \epsilon) + 1 \\ e & \text{if } x = n(n_0, \epsilon) + m(m_0, \epsilon) + 2 \\ 2 & \text{if } x > n(n_0, \epsilon) + m(m_0, \epsilon) + 2 \end{cases}$$

The function $f$ is in $V_\epsilon$ because $\cos^2 n_0 > 1 - \epsilon$ and $\cos^2 m_0 < \epsilon$, and $d$ is a correct program for $f$. The function $g$ is in $V_\epsilon$ because $\cos^2 n(n_0, c) < \epsilon$ and $\cos^2 m(m_0, c) > 1 - \epsilon$, and $e$ is a correct program for $g$. The functions $f$ and $g$ are different since $f(n_0 + 1) = 1$ and $g(n_0 + 1) = 0$ (since $n(n_0, c) > n_0$).

On the other hand, the finite automaton is not able to distinguish between $f$ and $g$. □

**Theorem 5.** *The class $V_\epsilon$ is not in $PrLFIN(\frac{1}{2} + \delta)$ for any $\delta > 0$.*

*Proof.* Assume to the contrary that there is a $\delta > 0$ and a probabilistic inductive inference machine $M$ with a finite memory such that $V_\epsilon$ is in $PrLFIN(\frac{1}{2} + \delta)$. Denote by $s$ the number of the internal states of $M$. Denote by $f[b]$ the initial fragment $f(0)$, $f(1)$, $f(2)$, $\cdots$, $f(b)$ of the target function $f \in V_\epsilon$. Let $f[b]$ and $g[p]$ be two initial fragments of functions in $V_\epsilon$. Adapting an argument of Rabin [Rab63], we consider the probabilities $\xi_1$, $\xi_2$, $\xi_3$, $\cdots$, $\xi_s$, to enter the states 1, 2, 3, $\cdots$, $s$, after processing the initial fragment $f[b]$ of the target function $f$. We consider also the probabilities $\zeta_1$, $\zeta_2$, $\zeta_3$, $\cdots$, $\zeta_s$, to enter the states 1, 2, 3, $\cdots$, $s$, after processing the initial fragment $g[p]$ of the target function $g$. Suppose that that $h(j)$, $h(j + 1)$, $\cdots$, $h(z)$, $h(z + 1)$, $h(z + 2)$, is a continuation of functions $f$ and $g$ such that the functions

$$F(x) = \phi_{h(z+1)} = \begin{cases} f(0) & \text{if } x = 0, \\ f(1) & \text{if } x = 1, \\ f(2) & \text{if } x = 2, \\ \cdots & \\ f(b) & \text{if } x = b, \\ h(j) & \text{if } x = b + 1, \\ h(j + 1) & \text{if } x = b + 2, \\ h(j + 2) & \text{if } x = b + 3, \\ \cdots & \\ h(z) & \text{if } x = z + b + 1 - j, \\ h(z + 1) & \text{if } x = z + b + 2 - j, \\ h(z + 2) & \text{if } x = 0z + b + 3 - j, \\ 2 & \text{if } x > 0z + b + 3 - j \end{cases}$$

$$G(x) = \phi_{h(z+2)} = \begin{cases} g(0) & \text{if } x = 0, \\ g(1) & \text{if } x = 1, \\ g(2) & \text{if } x = 2, \\ \cdots & \\ g(b) & \text{if } x = p, \\ h(j) & \text{if } x = p+1, \\ h(j+1) & \text{if } x = p+2, \\ h(j+2) & \text{if } x = p+3, \\ \cdots & \\ h(z) & \text{if } x = z+p+1-j, \\ h(z+1) & \text{if } x = z+p+2-j, \\ h(z+2) & \text{if } x = 0z+p+3-j, \\ 2 & \text{if } x > 0z+p+3-j \end{cases}$$

are in the class $V_\epsilon$. Denote by $\psi_1$, $\psi_2$, $\psi_3$, $\cdots$, $\psi_s$, the probability to output $h(z+1)$ if the inductive inference machine starts in the states 1, 2, 3, $\cdots$, $s$, respectively, and processes the fragment $h(j)$, $h(j+1), \cdots, h(z)$, $h(z+1)$, $h(z+2)$, of the target function. Then the probability to output the value $h(z+1)$ when processing the target function $F$ equals

$$\xi_1\psi_1 + \xi_2\psi_2 + \cdots + \xi_s\psi_s > \frac{1}{2} + \delta.$$

The probability to output the value $h(z+1)$ when processing the target function $G$ equals

$$\zeta_1\psi_1 + \zeta_2\psi_2 + \cdots + \zeta_s\psi_s < \frac{1}{2} - \delta.$$

By subtraction, we get

$$(\xi_1 - \zeta_1)\psi_1 + (\xi_2 - \zeta_2)\psi_2 + \cdots + (\xi_s - \zeta_s)\psi_s > 2\delta,$$

hence,

$$|\xi_1 - \zeta_1| + |\xi_2 - \zeta_2| + \cdots + |\xi_s - \zeta_s| > 2\delta.$$

We see that if the fragments $f[b]$ and $g[p]$ are distinguishable by a fragment $h(j)$, $h(j+1)$, $\cdots$, $h(z)$, $h(z+1)$, $h(z+2)$, then their vectors of probabilities $\xi_1$, $\xi_2$, $\xi_3$, $\cdots$, $\xi_s$, and, $\zeta_1$, $\zeta_2$, $\zeta_3$, $\cdots$, $\zeta_s$, are remote in a suitable metric of the $s$-dimensional space. Hence, at most $(1+\delta^{-1})^{n-1}$ pairwise indistinguishable fragments $f[b]$ are possible. (See the calculation in [Rab63]). The rest of our proof copies the proof of Theorem 4, only by $c$ we now denote the number $((1+\delta^{-1})^{n-1})!$ □

**Theorem 6.** *For arbitrary $\epsilon$ the class $V_\epsilon$ is in $QLFIN\langle 1-\epsilon\rangle$.*

*Proof.* The automaton has four non-halting non-output states $\{q_1, q_2, q_3, q_4\}$ (with $q_1$ as the initial state), four halting output states $\{q_6, q_7, q_9, q_10\}$ (when the current input value is output), and two non-halting non-output states $\{q_5, q_8\}$ used only at the very end of the inference.

When defining our class $V_\epsilon$ we denoted by $u$ the number of zero-values of the function in the class $V_\epsilon$, and we denoted by $v$ the number of one-values of the function.

We construct our automaton to have the following properties.

When the first value 'two' is read from the input, the distribution of the amplitudes for the non-halting states becomes

$$(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0, 0).$$

After $u$ zeros and $v$ ones are read from the input, the distribution of the amplitudes for the non-halting states becomes

$$(\frac{1}{\sqrt{2}}\cos u, \frac{1}{\sqrt{2}}\sin u, \frac{1}{\sqrt{2}}\cos v, \frac{1}{\sqrt{2}}\sin v, 0, 0, 0, 0, 0, 0).$$

After the reading the first input 'larger than two' the distribution of the amplitudes becomes

$$(0, 0, 0, 0, \frac{1}{\sqrt{2}}\cos u, \frac{1}{\sqrt{2}}\sin u, \frac{1}{\sqrt{2}}\cos v, \frac{1}{\sqrt{2}}\sin v, 0, 0).$$

At the states $q_6$ and $q_7$ the current input value $f(z)$ is output, at the states $q_5$ and $q_8$ no output is produced. After reading the second input 'larger than two', the states $q_5$ and $q_8$ are followed by the states $q_9$ and $q_{10}$, respectively, with output of the current input value $f(z+1)$ and termination of the inference process.

To get these properties, our automaton has the following unitary matrices corresponding to the input symbols 2, 0, 1, $L$ (where $L$ stands for 'larger than two'):

$$\begin{pmatrix}
\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

$$\begin{pmatrix}
\cos 1 & \sin 1 & 0&0&0&0&0&0&0&0 \\
\sin 1 & -\cos 1 & 0&0&0&0&0&0&0&0 \\
0 & 0 & 1&0&0&0&0&0&0&0 \\
0 & 0 & 0&1&0&0&0&0&0&0 \\
0 & 0 & 0&0&1&0&0&0&0&0 \\
0 & 0 & 0&0&0&1&0&0&0&0 \\
0 & 0 & 0&0&0&0&1&0&0&0 \\
0 & 0 & 0&0&0&0&0&1&0&0 \\
0 & 0 & 0&0&0&0&0&0&1&0 \\
0 & 0 & 0&0&0&0&0&0&0&1
\end{pmatrix}$$

$$\begin{pmatrix}
1&0 & 0 & 0 & 0&0&0&0&0&0 \\
0&1 & 0 & 0 & 0&0&0&0&0&0 \\
0&0 & \cos 1 & \sin 1 & 0&0&0&0&0&0 \\
0&0 & \sin 1 & -\cos 1 & 0&0&0&0&0&0 \\
0&0 & 0 & 0 & 1&0&0&0&0&0 \\
0&0 & 0 & 0 & 0&1&0&0&0&0 \\
0&0 & 0 & 0 & 0&0&1&0&0&0 \\
0&0 & 0 & 0 & 0&0&0&1&0&0 \\
0&0 & 0 & 0 & 0&0&0&0&1&0 \\
0&0 & 0 & 0 & 0&0&0&0&0&1
\end{pmatrix}$$

$$\begin{pmatrix}
0&0&0&0&1&0&0&0&0&0 \\
0&0&0&0&0&1&0&0&0&0 \\
0&0&0&0&0&0&1&0&0&0 \\
0&0&0&0&0&0&0&1&0&0 \\
0&0&0&0&0&0&0&0&1&0 \\
0&1&0&0&0&0&0&0&0&0 \\
0&0&1&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&1 \\
1&0&0&0&0&0&0&0&0&0 \\
0&0&0&1&0&0&0&0&0&0
\end{pmatrix}$$

It follows from the definition of the class $V_\epsilon$ that after the first value 'larger than 2' there follows another one. It follows also that either $(\cos u)^2 < \epsilon$ and $(\cos v)^2 > 1-\epsilon$, and $f(z)$ is a program to compute the function $f$, or $(\cos v)^2 < \epsilon$ and $(\cos u)^2 > 1-\epsilon$, and $f(z+1)$ is a program to compute the function $f$. □

## 6 Conclusions

Our paper relates both to learning theory and to quantum computation. We do not pretend to have discovered new effective machine learning algorithms performed by quantum computers. We were interested in theoretical capabilities and limitations of various learning models. It turns out that there are learning problems for which quantum algorithms have advantages over classical (deterministic or probabilistic) ones. Such advantages have already been discovered in papers on quantum computation [Shor94,AF98]. The results in this paper show

that quantum learning differs rather much from quantum computation. Quantum finite automata can recognize only languages recognizable by deterministic finite automata [KW97] but our results show there exist classes learnable by quantum finite automata but not learnable by deterministic finite automata. It may seem that our Theorem 5 is based on using quantum finite automata with very special parameters. What happens if a practical implementation of such an automaton has a slight error in the parameters? A more careful analysis shows that almost always quantum automata have advantages over their deterministic counterparts. It was essential in our Theorem 5 that the angle used in our matrices for the input symbols 0 and 1 is irrational with respect to $\pi$. However this is true for nearly all possible angles (the probability to have such an angle is 1, the probability to have angle which is not irrational with respect to $\pi$ equals 0).

## References


[AF98] A. Ambainis and R. Freivalds. *1-way quantum finite automata: strengths, weaknesses and generalizations.* Proc. 39th Symp. on Foundations of Computer Science, Palo Alto, November 8-11, 1998, IEEE Computer Society, pp. 332-341.

[AH87] S. Arikawa and M. Haraguchi. *A Theory of Analogical Reasoning.* Ohm-sha, 1987.

[AN91] S. Arikawa and T. Nishino. *Computational Approaches to Machine Learning* (in Japanese), JIPS, 32, 3, pp.217-225(1991-03).

[AM95] S. Arikawa and Y. Mukouchi. *Towards a mathematical theory of machine discovery from facts.* Theoretical Computer Science, vol. 137, 1995. pp. 53-84.

[BFS96] J. Bārzdiņš, R. Freivalds and C. H.Smith. *Learning with confidence.* Lecture Notes in Computer Science 1046, Springer, 1996, pp. 207-218.

[BFS98] J. Bārzdiņš, R. Freivalds and C. H. Smith. *A Logic of Discovery.* Lecture Notes in Artificial Intelligence 1532, Springer, 1998, pp. 401-402.

[BB75] L. Blum and M. Blum. *Toward a mathematical theory of inductive inference.* Information and Control, 1975, v.28, No. 2, pp. 125-155.

[BSS89] L. Blum, M. Shub and S. Smale. *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines.* Bulletin of the American Mathematical Society, v. 21, 1989, pp. 1-46.

[Fre91] R. Freivalds. *Inductive inference of recursive functions: qualitative theory.* Lecture Notes in Computer Science 502, Springer, 1991, pp. 77-110.

[FS92] R. Freivalds and C. H.Smith. *Memory limited inductive inference machines.* Lecture Notes in Computer Science 621, Springer, 1992, pp. 19-29.

[FKS95] R. Freivalds, E. B. Kinber and C. H. Smith. *On the impact of forgetting on learning machines.* Journal of the ACM, v.42, No.6, 1995, pp. 1146-1168.

[FKS95a] R. Freivalds, E. B. Kinber and C. H. Smith. *On the intrinsic complexity of learning.* Information and Computation, v.123, No 1, 1995, pp. 64-71.

[Gold67] E. M. Gold. *Language identification in the limit.* Information and Control, v. 10, 1967, pp. 447-474.

[KW97] A. Kondacs and J. Watrous. *On the power of quantum finite state automata.* Proc. 38th IEEE Conference on Foundations of Computer Science, 1997, pp. 66-75.

[MC00] C. Moore and J. Crutchfield. *Quantum automata and quantum grammars.* Theoretical Computer Science, v.237, 2000, pp. 275-306.

[Niv67] I. Niven. *Irrational Numbers.* The Carus Mathematical Monographs, vol. 11. The Mathematical Association of America, 1967.

[Pitt89] L. Pitt. *A characterization of probabilistic inference.* Journal of the ACM, v.36, No.2, 1989, pp. 383-433.

[PS88] L. Pitt and C. H. Smith. *Probability and plurality for aggregations of learning machines.* Information and Computation, v.77, No 1, 1988, pp. 77-92.

[Rab63] M. O. Rabin, *Probabilistic automata.* Information and Control, v. 6, No. 3, 1963, pp. 230-245.

[Rog87] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* MIT Press, 1987.

[Shor94] P. W. Shor. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM Journal of Computing, v. 26, No. 5, 1997, pp. 1484-1509.

[Wieh76] R. Wiehagen. *Limes-Erkennung rekursiver Funktionen durch spezielle Strategien.* J. Information Processing and Cybernetics (EIK), v. 12, 1976, pp. 93-99.

[WZ94] R. Wiehagen and T. Zeugmann. *Ignoring Data May be the Only Way to Learn Efficiently.* Journal of Experimental and Theoretical Artificial Intelligence, v. 6, No. 1, 1994, pp. 131-144.

# Capabilities of Finite Standardization

Gints Tervits*

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia

**Abstract.** It was presumed that finite standardizability is a type of inductive inference only slightly more powerful than finite identifiability. Two theorems are proved to show that finite standardizability is, quite to the contrary, only slightly less powerful than identifiability in the limit.

## 1 Introduction

Inductive inference is the term used for reconstruction of programs from sample computations. It is the part of computational learning theory heavily based on the recursive functions theory. Started by the well-known paper [Go 67] nowadays inductive inference is the most developed part of computational learning theory (see the survey [AS 83] and the monograph [JORS 99]). Far from development of effective machine learning algorithms, inductive inference has actively supplied the practitioners with ideas on how to learn, what to learn, what difficulties to avoid.

The most widely used inference type is $EX$ (explanatory identification).

**Definition 1.** *Let $\mathcal{U}$ be a class of total recursive functions and $\varphi$ be a Gödel numbering of all 1-argument partial recursive functions. $\mathcal{U}$ is called identifiable in the limit ($\mathcal{U} \in EX$) if and only if there is a partial recursive strategy $S$ such that for arbitrary $f \in \mathcal{U}$, there is $i \in N$ such that $\varphi_i = f$ and $S(f^{[n]}) = i$ for almost all $n \in N$.*

Gold [Go 67] proved that the class $\mathcal{R}$ of all the total recursive functions is not identifiable in the limit ($\mathcal{R} \notin EX$). Since then many generalizations and restrictions of $EX$-identifiability have been considered, see [WZ 95] for a survey. The most popular restriction of $EX$ is finite identifiability.

**Definition 2.** *Let $\mathcal{U}$ be a class of total recursive functions and $\varphi$ be a Gödel numbering of all 1-argument partial recursive functions. $\mathcal{U}$ is called finitely identifiable ($\mathcal{U} \in FIN$) if and only if there is a recursive functional $F$ on $\mathcal{U}$ such that $\varphi_{F(f)} = f$ for arbitrary $f \in \mathcal{U}$.*

* Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission, and the Swedish Institute, Project ML-2000

Studying inductive inference of minimal programs Freivalds [Fr 75] found that a certain criterion becomes simple and natural if we consider counterparts for $FIN$ (where the recursive functional is replaced by an effective operation), and $EX$ (where the strategy working in the limit is replaced by a limit-effective operation).

**Definition 3.** *A functional $F$ on $\mathcal{U}$ (of $\mathcal{U} \to N$ type) is called effective operation if there is a partial recursive function $\psi$ such that, for arbitrary $\varphi_x \in \mathcal{U}$,*

1. *$F(\varphi_x)$ defined $\Leftrightarrow$ $\psi(x)$ defined,*
2. *$F(\varphi_x)$ defined $\Rightarrow$ $(F(\varphi_x) = \psi(x))$.*

**Definition 4.** *A class $\mathcal{U}$ of total recursive functions is called finitely standardizable ($\mathcal{U} \in FSTAND$) if and only if there is an effective operation $F$ on $\mathcal{U}$ such that, for arbitrary $f \in \mathcal{U}$, $\varphi_{F(f)} = f$.*

It is easy to see that $\mathcal{U} \in FIN$ implies $\mathcal{U} \in FSTAND$. There is a classical theorem suggesting the reverse also might take place.

**Theorem 1.** [Kreisel, Lacombe, Shoenfield [KLS 57]] *$F$ is a total effective operation on $\mathcal{R}$ if and only if $F$ is a total recursive functional on $\mathcal{R}$.*

Since standardizability is performed by a total effective operation, one might suggest that $FSTAND = FIN$. However, $FSTAND \neq FIN$, as it was proved in [FKW 84]. Since then the notion of standardizability was used by many authors, but it was always supposed that $FSTAND$ is only 'slightly' larger than $FIN$. We disprove this belief in this paper.

Constructive ordinals were used as a measure of the number of mindchanges in the $EX$-identification process. The precise definition is too long for this paper and we refer to [FS 93]. The idea however is simple. The inductive inference machine has a special 'counter' containing arbitrary constructive ordinal (for instance, arbitrary natural number, since natural numbers are the smallest ordinals). At every mindchange the content of the counter is diminished, and after a finite number of steps the counter can become empty. After that moment new mindchanges are not allowed.

Nearly all $EX$-identifiable classes of total recursive functions are also $EX$-identifiable with an ordinal-bounded mindchange complexity. Hence only complicated classes are $EX$-identifiable without any ordinal bound. We have managed to prove that there is a finitely standardizable class of total recursive functions which is very complicated in this sense, namely, it cannot be $EX$-identified with an ordinal-bounded number of mindchanges. However every finitely standardizable class is $EX$-identifiable.

The notions and notation not explained here can be found in [Ro 67].

## 2 Results

Freivalds, Alberts, and Smith [FAS96] proved the following theorem.

**Theorem 2.** **[FAS96]** *If a class $\mathcal{U}$ of total recursive functions is finitely standardizable, then $\mathcal{U}$ is identifiable in the limit.*

*Proof.* Let $\psi$ be the partial recursive function defining an effective operation $F$ such that, for arbitrary $f \in \mathcal{U}$, $(F(f) = u)$ implies $(\varphi_u = f)$. We fix a concrete procedure of computation of $\psi$, and this allows us to describe the number of steps of the computation for specific values $\psi(x)$. We use $Dom\ \chi$ to denote the domain of a function $\chi$.

The following instructions describe how to compute an auxiliary partial recursive function $\eta(x, y, z, k)$.

Compute $\psi(y)$. If $\psi(y)$ is not defined, then $\eta(x, y, z, k)$ is not defined as well. If $\psi(y)$ is computed, then perform $x$ steps of computation to compute $\psi(z)$. If $\psi(z)$ is not computed in $x$ steps, then $\eta(x, y, z, k) = \varphi_y(x)$. If $\psi(z)$ is computed in at most $x$ steps, then check whether or not $\psi(z) = \psi(y)$. If the equality does not hold, then define $\eta(x, y, z, k) = \varphi_y(x)$. If the equality holds then denote the number of steps for the computation of $\psi(z)$ by $w$ (please notice that $w < x$); compute the values of $\varphi_y$ and $\varphi_k$ until $\varphi_y(t)$ and $\varphi_k(t)$ are computed for all $t \leq w$ (if this never happens, then $\eta(x, y, z, k)$ is not defined). After that check whether or not $\varphi_y^{[w]} = \varphi_k^{[w]}$. If they are not equal, then $\eta(x, y, z, k)$ is not defined. If they are equal, then $\eta(x, y, z, k) = \varphi_k(x)$.

Using the s-m-n theorem [Ro 67] we obtain a total recursive function $h(y, z, k)$ such that $\varphi_{h(y,z,k)}(x) = \eta(x, y, z, k)$. We have:

$$\varphi_{h(y,z,k)} = \begin{cases} \text{nowhere} \\ \text{defined} & \text{, if } \psi(y) \text{ is not defined;} \\ \varphi_y & \text{, if } \psi(y) \text{ is defined, and } \psi(z) \text{ is not defined;} \\ \varphi_y & \text{, if } \psi(y) \text{ and } \psi(z) \text{ are defined, and } \psi(y) \neq \psi(z); \\ \varphi_k & \text{, if } \psi(y) \text{ and } \psi(z) \text{ are defined, } \psi(z) \text{ is computed in } w \text{ steps, } \psi(z) = \psi(y), \{0, 1, 2, \ldots, w\} \subseteq Dom\ \varphi_y, \text{ and } \varphi_y^{[w]} = \varphi_k^{[w]}; \\ \varphi_y^{[w]} & \text{, otherwise} \end{cases}$$

By the recursion theorem [Ro 67], there is a total recursive function $n(y,k)$ such that:

$$\varphi_{n(y,k)} = \begin{cases} \text{nowhere defined} & \text{, if } \psi(y) \text{ is not defined;} \\ \varphi_y & \text{, if } \psi(y) \text{ is defined, and either } \psi(n(y,k)) \text{ is not defined, or } \psi(n(y,k)) \neq \psi(y); \\ \varphi_k & \text{, if } \psi(y) \text{ and } \psi(n(y,k)) \text{ are defined, } \psi(n(y,k)) \text{ is computed in } w \text{ steps, } \psi(n(y,k)) = \psi(y), \{0,1,2,\ldots,w\} \subseteq \mathit{Dom}\ \varphi_y, \text{ and } \varphi_y^{[w]} = \varphi_k^{[w]}; \\ \varphi_y^{[w]} & \text{, otherwise (where } w \text{ is the number of steps in computation of } \psi(z)). \end{cases}$$

By $w'(y,k)$ we denote the precise number of steps of computation of $\psi(n(y,k))$. We notice some properties of $n(y,k)$ and $w'(y,k)$.

First, if $\varphi_y$ is a total recursive function in $\mathcal{U}$, and $\psi(y)$ is defined, then for arbitrary $k$, $w'(y,k)$ is defined (otherwise $\varphi_{n(y,k)}$ would have been equal $\varphi_y$, but $\psi(y)$ is defined and $\varphi_y \in \mathcal{U}$) and $\psi(n(y,k)) = \psi(y)$ (the same argument).

Second, if $\varphi_y$ is a partial recursive function such that $\psi(y)$ is defined, $\psi(n(y,k))$ equals $\psi(y)$, and $\varphi_k$ is a total recursive function in the class $\mathcal{U}$ such that $\varphi_k^{[w'(y,k)]}$ equals $\varphi_y^{[w'(y,k)]}$, then $\psi(k)$ is defined, and $\psi(k) = \psi(y)$ (in this case $\varphi_{n(y,k)}$ equals $\varphi_k$, and $\varphi_k \in \mathcal{U}$, hence $\psi(n(y,k)) = \psi(k)$, and $\psi(n(y,k)) = \psi(y)$).

Put $w(y,v) = \max_{k \leq v} w'(y,k)$. It follows by the above properties ('first' and 'second') that:

A) for arbitrary $y$, if $\varphi_y$ is a total recursive function in $\mathcal{U}$, and $\psi(y)$ is defined, then, for arbitrary $k$ and $v$, the values $\psi(n(y,k))$ and $w(y,v)$ are defined, and $\psi(n(y,k)) = \psi(y)$;

B) for arbitrary $y, v$, if $\varphi_y$ is a partial recursive function such that $\psi(y)$ and $w(y,v)$ are defined, and for arbitrary $s \leq v$, $\psi(n(y,s)) = \psi(y)$ holds, and if $k \leq v$ and $\varphi_k$ is a function in the class $\mathcal{U}$ such that $\varphi_k^{[w(y,v)]} = \varphi_y^{[w(y,v)]}$, then $\psi(k) = \psi(y)$.

We now define a recursively enumerable set $\mathcal{M}$ of triples of integers. The triple $\langle a,b,c\rangle$ is in $\mathcal{M}$ if there is a $y$ such that $\psi(y) = c$, $w(y,b)$ is defined, $\varphi_y(0)$, $\varphi_y(1)$, $\ldots$ $\varphi_y(w(y,b))$ are defined, for arbitrary $s \leq b$, the equality $\psi(n(y,s)) = \psi(y) = c$ holds, and $a$ equals the canonical index of the graph of $\varphi_y^{[w(y,b)]}$.

We will prove that for arbitrary $f \in \mathcal{U}$ and for arbitrary natural number $c$:

1. $(F(f) = c) \Rightarrow [(\forall b)(\exists a)(\exists x)(\langle a, b, c\rangle \in \mathcal{M}\ \&\ a = \tau f^{[x]})]$,
2. $[(\exists b)\ (\exists a)\ (\exists x)\ (\langle a, b, c\rangle \in \mathcal{M}\ \&\ b \leq \min_{\varphi}(f)\ \&\ a = \tau f^{[x]})] \Rightarrow F(f) = c$.

We prove 1). Let $\varphi_y$ be a function in $\mathcal{U}$, and $F(\varphi_y) = c$, i. e. $\psi(y) = c$. It follows from A) that, for arbitrary $v, k$, the values $w(y, k)$ and $\psi(n(y, k))$ are defined, and $\psi(n(y, k)) = \psi(y) = c$. By the definition of $\mathcal{M}$, for arbitrary $b$, the set $\mathcal{M}$ contains the triple $\langle a, b, c\rangle$, where $a$ is the canonical index of the graph of $\varphi_y^{[w(y,b)]}$.

We prove 2). Let the triple $\langle a, b, c\rangle$ be in $\mathcal{M}$, $k \leq b$, $\varphi_k \in \mathcal{U}$, and $a$ be a canonical index of the graph of some initial fragment of the function $\varphi_k$. Then by the definition of $\mathcal{M}$, there is a $y$ such that $\psi(y) = c$, $w(y, b)$ is defined, $\varphi_y(0), \varphi_y(1), \ldots, \varphi_y(w(y, b))$ defined, and, for arbitrary $s \leq b$ the equalities $\psi(n(y, s)) = \psi(y) = c$ and $\varphi_y^{[w(y,b)]} = \varphi_k^{[w(y,b)]}$ hold. Then it follows from B) that $\psi(k) = c$, i. e. $F(\varphi_k) = c$.

We define an auxiliary recursive functional $F'(f, t)$ on $\mathcal{U} \times N$. For computation of $F'(f, t)$ enumerate $\mathcal{M}$ until a triple $\langle a, t, c\rangle$ is found where $a$ is canonical index of an initial fragment of the function $f$, and $c$ is an integer. Then we define $F'(f, t)$. It follows from 1) and 2) that, for arbitrary $f \in \mathcal{U}$ and arbitrary $t \geq \min_{\varphi}(f)$, the equality $F'(f, t) = F(f)$ holds.

Finally, for the identification in the limit of arbitrary $f \in \mathcal{U}$, follow the subsequent instructions. To output the $t$-th hypothesis, compute in parallel $F'(f, z)$ for various $z \geq t$, and output the first obtained value. It is easy to see that at most $\min_{\varphi}(f)$ hypotheses can differ from $F(f)$. □

This theorem was supplemented in [FAS96] by the following theorem.

**Theorem 3.** *There is a class $\mathcal{U}$ of total recursive functions such that:*

1. *$\mathcal{U}$ is finitely standardizable,*
2. *the number of mindchanges to identify the class $\mathcal{U}$ in the limit cannot be restricted by any fixed constant.*

We strengthen this in the following way.

**Theorem 4.** *There is a class $\mathcal{U}$ of total recursive functions such that:*

1. *$\mathcal{U}$ is finitely standardizable,*
2. *$\mathcal{U}$ is not identifiable in the limit with any ordinal-bounded mindchange complexity.*

*Proof.* A non-empty class $\mathcal{U}$ of total recursive functions and effective operation on $\mathcal{U}$ will be constructed such that:

a) for arbitrary $f \in \mathcal{U}$, the value of $F(f)$ is a correct $\varphi$-index of the function $f$,
b) for arbitrary initial fragment $f^{[x]}$ of an arbitrary function $f \in \mathcal{U}$ there is a different function $g \in \mathcal{U}$ with the same initial fragment $g^{[x]} = f^{[x]}$.

1. We prove that the assertion 2) of the theorem is implied by a) and b).

Assume that $\mathcal{U}$ is identifiable in the limit with $\alpha$ mindchanges (where $\alpha$ is a constructive ordinal) and the identification is performed by an inductive inference machine $\mathcal{M}$. Take an arbitrary $f_0 \in \mathcal{U}$ and consider the performance of $\mathcal{M}$ when processing $f_0$. At some moment $\mathcal{M}$ produces the first hypothesis $h_1$ on $f_0$. The hypothesis was produced by $\mathcal{M}$ based only on a finite initial fragment $f_0^{[x]}$ of $f_0$. At this moment the ordinal $\alpha$ is diminished and substituted by an $\alpha_1 < \alpha$. It follows from b) that there is a function $f_1 \in \mathcal{U}$ such that $f_1^{[x]} = f_0^{[x]}$ but the first hypothesis $h_1$ is not a correct $\varphi$-index for $f_1$.

Now consider the performance of $\mathcal{M}$ when processing $f_1$. The machine $\mathcal{M}$ produces the first hypothesis $h_1$ (based on the same $f_1^{[x]} = f_0^{[x]}$) and later (since $\mathcal{M}$ is correct on $f_1$) $\mathcal{M}$ produces the second hypothesis $h_2$ (based on a finite initial fragment $f_1^{[y]}$), and at this moment the ordinal $\alpha_1$ is diminished and substituted by an $\alpha_2 < \alpha_1$. It follows from b) that there is a functional $f_2 \in \mathcal{U}$ such that $f_2^{[y]} = f_1^{[y]}$ but the second hypothesis $h_2$ is not a correct $\varphi$-index for $f_2$.

Continuing this argument we get arbitrarily long sequence of descending constructive ordinals $\alpha, \alpha_1, \alpha_2, \ldots$. However this contradicts the essential properties of ordinals. This completes the proof of 2).

2. We construct the class $\mathcal{U}$ and the effective operation $F$ with the properties a), b). Every function in the class $\mathcal{U}$ is a function differing from some constant only for a finite number of values of the argument.

We construct simultaneously the class $\mathcal{U}$ and a recursively enumerable set $T$ of pairs $\langle a, b\rangle$ (to be used for the definition of the effective operation). In the process of the construction some functions will be placed into an auxiliary class $\mathcal{U}'$. If a function gets into $\mathcal{U}'$, it can be removed from $\mathcal{U}'$ or it can stay in $\mathcal{U}'$ forever. If a function is removed, it never returns to $\mathcal{U}'$. The class $\mathcal{U}$ consists of all the functions such that they stay in $\mathcal{U}'$ forever. By $a_0 a_1 \ldots a_n a^\infty$ we denote the function

$$f(x) = \begin{cases} a_0 & \text{, if } x = 0; \\ a_1 & \text{, if } x = 1; \\ \cdots & \\ a_n & \text{, if } x = n; \\ \alpha & \text{, if } x > n. \end{cases}$$

By $\{a_0 a_1 \ldots a_n\}$ we denote the canonical index of the string $a_0 a_1 \ldots a_n$. By $s(a_0 a_1 \ldots a_n)$ we denote a $\varphi$-index of the function

$$g(x) = \begin{cases} a_0 & \text{, if } x = 0; \\ a_1 & \text{, if } x = 1; \\ \cdots & \\ a_n & \text{, if } x = n; \\ a_n & \text{, if } x > n. \end{cases}$$

obtained from $\{a_0 a_1 \ldots a_n\}$ by usage of some fixed uniform procedure.

The construction of $\mathcal{U}$ and $T$ is organized in stages.

STAGE 0. The functions $0^\infty$ and $1^\infty$ are placed in $\mathcal{U}'$ and the pairs $\langle\{0\},0\rangle$, $\langle\{1\},0\rangle$ are placed into $T$. Go to Stage 1.

STAGE $n+1$. Assume by induction that every pair $\langle a,b\rangle$ placed into $T$ at Stage $n$ is of form $\langle\{a_0a_1\dots a_n\},n\rangle$ and it corresponds to a function $a_0a_1\dots a_na_n^\infty$ placed into $\mathcal{U}'$. We also assume by induction that at the end of Stage $n$ $\mathcal{U}'$ does not contain any functions different from those corresponding to pairs placed into $T$ at Stage $n$.

We compute

$$\begin{aligned}&\varphi_0(0)\\ &\varphi_1(0),\ \varphi_1(1)\\ &\dots\\ &\varphi_n(0),\ \varphi_n(1),\ \dots,\ \varphi_n(n)\end{aligned}$$

$n$ steps of computation each. After that we consider all the pairs placed into $\mathcal{U}'$ at Stage $n$. When considering the pair $\langle\{a_0a_1\dots a_n\},n\rangle$ we test whether or not there is a $k\le n$ such that:

(i) computation of all $\varphi_k(0)$, $\varphi_k(1),\dots,$ $\varphi_k(k)$ terminates in at most $n$ steps,
(ii) $\varphi_k(0)=a_0$, $\varphi_k(1)=a_1$, …, $\varphi_k(k)=a_k$,
(iii) $(\exists l)((k\le l\le n)\ \&\ (a_{i-1}\ne a_i))$

If such a $k$ exists, then the function $a_0a_1\dots a_na_n^\infty$ is removed from $\mathcal{U}'$. If such a $k$ does not exist, the function $a_0a_1\dots a_na_n^\infty$ remains in $\mathcal{U}'$, and we place the following pairs into $T$:

$$\begin{aligned}&\langle\{a_0a_1\dots a_na_n\},n+1\rangle\\ &\langle\{a_0a_1\dots a_n(a_n+1)\},n+1\rangle\\ &\langle\{a_0a_1\dots a_n(a_n+2)\},n+1\rangle\\ &\dots\\ &\langle\{a_0a_1\dots a_n(a_n+n+3)\},n+1\rangle,\end{aligned}$$

and we additionally place the following $n+3$ functions into $\mathcal{U}'$:

$$\begin{aligned}&a_0a_1\dots a_n(a_n+1)^\infty\\ &a_0a_1\dots a_n(a_n+2)^\infty\\ &\dots\\ &a_0a_1\dots a_n(a_n+n+3)^\infty\end{aligned}$$

Go to Stage $(n+2)$.

Now we prove that the class $\mathcal{U}$ is nonempty. Indeed, the functions $0^\infty$ and $1^\infty$ cannot be removed from $\mathcal{U}'$ because of (iii).

We define the effective operation by defining a partial recursive function $\psi(x)$. To compute this value, in parallel generate the pairs in $T$, and compute $\varphi_x(0),\varphi_x(1),\dots$. Let the pair

$$\langle\{a_0a_1\dots a_n\},b\rangle\in T$$

be the first pair (according to our parallel computation) such that $\varphi_x^{[n]}=a_0a_1\dots a_n$ and $x\le b$. Let $m$ be the largest integer $m\le n$ such that $a_m=a_{m+1}=\dots=a_n$. Then we define $\psi(x)=s(a_0a_1\dots a_m)$.

Now we prove that $\psi$ defines an effective operation on $\mathcal{U}$. (Please notice that $\psi$ does not define an effective operation on $\mathcal{R}$.)

Let $\varphi_x \in \mathcal{U}$. If $\varphi_x$ is a constant function $\varphi_x(t) = a$ then, by our definition, $\psi(x) = s(a)$, i. e. $\psi(x)$ does not depend on the particular $\varphi$-index of the function. If $\varphi_x$ is a non-constant function, then $\varphi_x = a_0a_1 \dots a_n a_{n+1}^{\infty}$. Assume that in this notation $n$ is chosen such that $a_n \neq a_{n+1}$. In this case $x \geq n+1$ (otherwise the function would have been removed from $\mathcal{U}'$). However for all $b \geq n+1$ the pair $\langle \{a_0a_1 \dots a_b\}, b\rangle$ compatible with the function $\varphi_x$ leads to representation

$$a_0a_1 \dots a_b = a_0a_1 \dots a_n a_{n+1} \dots a_b$$

with $a_{n+1} = a_{n+2} = \dots = a_b$. This implies $\psi(x) = s(a_0a_1 \dots a_n)$.

Now we prove the property b) of the class $\mathcal{U}$. Let the function $f = a_0a_1 \dots a_n a_{n+1}^{\infty}$ be in $\mathcal{U}$ ($a_n \neq a_{n+1}$), and let $f^{[x]}$ be the initial fragment from the property b). If $x \geq n+1$, then at Stage $x$ the functions

$$\begin{aligned} &a_0a_1 \dots a_n a_{n+1} \dots a_{x-1}(a_x+1)^{\infty} \\ &a_0a_1 \dots a_n a_{n+1} \dots a_{x-1}(a_x+2)^{\infty} \\ &\dots \\ &a_0a_1 \dots a_n a_{n+1} \dots a_{x-1}(a_x+x+2)^{\infty} \end{aligned}$$

are placed into $\mathcal{U}'$.

At least one of these $x+2$ functions is such that its minimum $\varphi$-index exceeds $x+1$. This function $g$ is different from $f$, and it is never removed from $\mathcal{U}'$. Hence this function is in $\mathcal{U}$, and $g^{[x]} = f^{[x]}$.

If $x < n+1$, then consider the function $f = a_0a_1 \dots a_n a_{n+1}^{\infty} \in \mathcal{U}$ and the functions

$$\begin{aligned} &a_0a_1 \dots a_x(a_x+1)^{\infty} \\ &a_0a_1 \dots a_x(a_x+2)^{\infty} \\ &\dots \\ &a_0a_1 \dots a_x(a_x+x+3)^{\infty} \end{aligned}$$

placed into $\mathcal{U}'$ at Stage $x+1$. At least one of these $x+3$ functions is such that its minimum $\varphi$-index is at least $x+1$. Hence this function $g$ is never removed from $\mathcal{U}'$, and $g^{[x]} = f^{[x]}$. □

## References


[AS 83] D. Angluin and C. H. Smith. *Inductive inference: theory and methods,* Computing Surveys, v. 15, 1983, pp. 237-269.

[BFS00] J. Bārzdiņš, R. Freivalds and C. H. Smith. *Towards a logic of discovery.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning. Proc. Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 110-120.

[BFS01] J. Bārzdiņš, R. Freivalds and C. H. Smith. *Towards axiomatic basis of inductive inference.* Lecture Notes in Computer Science 2138, 2001, pp. 1-13.

[Fr 75] R. Freivalds. *Minimal Gödel numbers and their identification in the limit.* Lecture Notes in Computer Science 32, 1975, pp. 219-225.

[Fr 78] R. Freivalds. *Effective operations and functionals computable in the limit.* Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, v. 24, 1978, pp. 193-206 (in Russian)

[FAS96] R. Freivalds, M. Alberts and C. H.Smith. *Finite standardizability characterized in identification complexity terms.* Proc. 6-th Int. Conf. "Information Processing and Management of Uncertainty in Knowledge-based Systems", Granada, Spain, July 1996, vol. 3, pp. 1399-1403.

[FS 93] R. Freivalds and C. H. Smith. *The role of procrastination in machine learning.* Information and Computation, vol. 107, 1993, pp. 237-271.

[FKW 84] R. Freivalds, E. B. Kinber and R. Wiehagen. *Connections between identifying functionals, standardizing operations and computable numberings.* Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, v. 30, 1984, pp. 145-164.

[Go 67] E. M. Gold. *Language identification in the limit.* Information and Control, v. 10, 1967, pp. 447-474.

[JORS 99] S. Jain, D. N. Osherson, J. S. Royer and A. Sharma. *Systems that Learn.* MIT Press, 1999.

[KLS 57] G. Kreisel, D. Lacombe and J. R. Shoenfield. *Partial recursive functionals and effective operations.* In: A. Heyting (ed.), Constructivity in mathematics: proceedings of the colloquium held at Amsterdam, North-Holland, 1957, pp. 195-207.

[Ro 67] H. Rogers Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.

[WZ 95] R. Wiehagen and T. Zeugmann. *Learning and Consistency.* Lecture Notes in Computer Science 961, 1995, pp. 1-24.

# A Survey of Quantum Learning

Richard Bonner[1] and Rūsiņš Freivalds[2]

[1] Department of Mathematics and Physics
Mälardalen University, Västerås, Sweden***.
rbr@mdh.se
[2] Department of Computer Science,
University of Latvia, Riga, Latvia†.
richard.bonner@mdh.se
rusins@cclu.lv

**Abstract.** We survey papers on problems of learning by quantum computers. The quest of quantum learning, as that of quantum computation, is to produce tractable quantum algorithms in situations, where tractable classical algorithms do not exist, or are not known to exist. We see essentially three papers [18, 92, 93], which in this sense separate quantum and classical learning. We also briefly sample papers on quantum search, quantum neural processing, and quantum games, where quantum learning problems are likely to appear.

Key words: *quantum learning, quantum search, quantum neural processing, quantum decisions, quantum games*

## 1 Introduction

It is not unnatural, at least historically, to see learning as a quantification of feedback - a famous concept, so eloquently launched by Wiener [121] half a century ago, on which the edifice of Cybernetics was to be constructed. Wiener's ideas, recall, concerned the behavior, of 'man' and 'machine' alike, and specifically its dynamic improvement by interactive evaluation. In computational perspective, roughly, a learner is (a mathematical model of) a computer, a behavior is a mathematical object such as a computer program, and learning is an algorithm, which modifies the behavior. To evaluate the behavior, the learner is placed more or less explicitly in a context of an economic game, a learning environment.

Clearly, learning depends on the underlying model of computation. In recent years, computation by quantum physical systems has been recognized as a phenomenon of fundamental scientific interest with promising pragmatic potential; see any of the introductory texts [50, 53, 66, 69, 83]. Hence the interest of quantum learning. The ultimate quest here is to produce tractable quantum algorithms in situations, where tractable classical learning algorithms do not

*** Supported by the ML2000 project sponsored by the Swedish Institute.
† Research supported by the Latvian Council of Science, grant 01.0354; European Commission, contract IST-1999-11234; Swedish Institute, project ML2000

exist, or are not known to exist. Welcome side effects may be expected from unavoidable formal revision of learning models to deal with the often un-intuitive quantum case.

Quantum learning is a theory in the making and its scientific production is rather fragmented. To set a background for its discussion let us symbolically recall some basic notions of classical learning and the intuition behind them.

An agent feels its environment through its sensors, a collection of functions of the state of the world. For any value of its sensors, the agent computes a response - a state of its effectors. The computation produces an input-output function $f$; this is the agent's behavior. For example, if the admissible output is a single binary variable, we talk of the agent making a decision or recognizing a concept. We could equate the agent's behavior with its knowledge, though this usually entails considerations of the internal implementation of $f$.

The agent is capable of different behaviors $f_\alpha$, some better than others. For example, a behavior could be determined by a neural network, the parameters $\alpha$ expressing the weights of neural connections and neural threshold values. Or, it could be computed by a $\text{program}_\alpha$ run on a classical or quantum computer. The agent learns (or is being trained) by successively improving its behavior through feed-back until satisfactory (target) behavior is reached. The problem of learning (for us, who design and train the agent) is to ascertain which behaviors the agent is capable of, to evaluate behaviors, and to see whether satisfactory behavior can effectively be learned under specified conditions; clearly, the balance of the value of behavior against its learning costs is crucial here.

Learning theory is thus in broad perspective a border area of computer science and economics, concerned with effective construction of strategies for (or by) economic agents, where the net value of a strategy includes all computation and information costs. This concept of learning is approached from the side of economics by learning in games [44], and from the side computer science by reinforcement learning [101]. A related approach incorporates learning into decision processes [17, 43, 102, 120]. Other models of learning in economics are surveyed in [100]. One may also mention machine learning [73], part of the field of artificial intelligence, which falls into this general category but is largely an empirical science.

In a first approximation, however, it is not unnatural to separate economics and computing. Economic theory then provides target behaviours, and the computational sciences study their learnability; it is then tacitly assumed that the more accurate the learning, the higher its economic value. Under this simplification, learning theory is about effective, exact or approximate, identification of a mathematical object by drawing on available data about the object; it is then a mathematical subject in its own rights and independent of its pragmatic roots. The usual duality considerations now enter.

In a computational approach, one may study algorithms, which construct a target object from data; this has the advantage that little *a priori* knowledge about the object is needed, the learning being considered successful if the output of an algorithm stabilizes. This approach has been formalized by Gold

[46] under the banner of identification in the limit. It is by necessity formal, almost exclusively used in theoretical work [59] and in formal applications such as logic programming [15]. It is akin to the conceptual framework of Kolmogorov-Solomonoff algorithmic information theory [68].

Alternatively, if *a priori* knowledge about the object is available, one may study how this knowledge increases in function of incoming object data, until the object has been identified with required accuracy. This is the information-theoretic approach, initiated by Shannon [97], now also known as the theory of search [2]. It uses probabilistic methods and often goes under the banner of statistical learning theory [107]. Not formally computational, it's practical application range is wide, intersecting with that of classical statistics [77].

Though pragmatically quite different, the two approaches are essentially dual to one another in a formal mathematical sense, for example, along the ideas of 'formal concept analysis' [45], modulo computational complexity of encoding objects and processing [7, 65]. It is consequently common practice to derive sample complexity bounds for the convergence of learning algorithms by information-theoretic methods. This is the point of departure of computational learning theory (CoLT), originally supplying sample bounds for neural network training [8, 64, 118]. It rests on the concept of probably approximate convergence (PAC) introduced into learning by Valiant [106], and builds on the theory of empirical processes [84] and uniform central limit theorems [33]. Here, the insistence on the uniformity of the convergence of learning with respect to the distribution of learning data reduces the *a priori* knowledge of the learning context, essentially to a specification of a class to which the target of learning is to belong.

## 2 Quantum learning today

None of the introductory texts [50, 53, 66, 69, 83] on quantum computation and information theory mentions quantum learning. We did find one book with 'quantum learning' in the title [86], but largely a popular discussion. At the time of writing, the search of the Los Alamos preprint archive at http://xxx.lanl.gov/archive/quant-ph/ for 'learning' returned five papers, and the NEC server http://citeseer.nj.nec.com/ cs returned thirteen papers to 'quantum learning'. Clearly, 'learning' is still a relatively new term in the world of quantum computation.

Before discussing the key papers, we note two early papers [28, 29] from 1995 and 1997 by Chrisley, non-mathematical discussions signalling an interest in quantum learning in cognitive science. We also note several papers [108–110, 113–115] by Ventura and Martinez from the years 1997-2000, investigating quantum learning, largely in connection with neurocomputing; of these, perhaps most interesting are [109, 110], in that they consider quantum learning (of DNF in $n$ Boolean variables) by *classical* example oracle (and claim learning time $O(2^{\frac{n}{2}})$).

There is also a paper [80] by Pearson *et al*, where classical learning in form of a genetic algorithm was employed to control a quantum physical system in laser photochemistry.

Few mathematical papers address quantum learning expressly. Apart from our paper [18] in the present collection, on identification in the limit by quantum finite automata, all such papers consider learning of Boolean functions with either a quantum membership oracle or a quantum example oracle. Quantum computation relative to oracles was taken up by Berthiaume and Brassard [16] already in 1992, but it appears the first to use quantum oracles in learning were Bshouty and Jackson (1995) [23, 24]. They defined a quantum extension QEX of the classical example oracle EX, and showed that the class DNF is *effectively* PAC learnable by a quantum Turing machine using QEX with uniform example distribution; this was interesting, as no classical such algorithm was known to exist (that the DNF class is classically efficiently learnable from membership queries was shown by Jackson [57] a year earlier). There is also a recent technical follow-up paper by Jackson *et al* [58], containing in particular a lower sample complexity bound for quantum PAC learning of DNF under the uniform distribution.

This work was recently put into broader perspective in the lucid paper [93] of Servedio and Gortler (2001), which we presently relate at some length. They consider two standard learning models of Boolean functions: exact learning from membership queries, due to Angluin [6], and PAC learning from examples, due to Valiant [106]. In both models, the point of departure is a class $F = \bigcup_n F_n$ of Boolean functions, $F_n \subset \{0,1\}^n$, $n \geq 1$, and a black-box (oracle) access to a fixed function $f \in F$. A learning algorithm for the class $F$ makes calls to the oracle, and, on the basis of received information, iteratively computes a hypothesis $h \in F$, which estimates $f$. The oracle could be classical (c) or quantum (q), and so could be the computation of $h$; hence, formally, four combinations are possible: c-c, c-q, q-c, and, q-q, say. Servedio and Gortler relate query complexity bounds for the c-c and the q-q algorithms for the Angluin and the Valiant models; of the remaining two cases, c-q is clearly more interesting, but it seems still uninvestigated (apart from the earlier mentioned papers [109, 110] of Ventura and Martinez).

Recall first the well-known c-c case. In the membership query model [6], a classical (probabilistic) learning algorithm $L$ accesses $f$ through a 'membership oracle' $MQ_f$, which, upon a query $x \in \{0,1\}^n$, returns the bit $f(x)$. The algorithm $L$ is said to learn $F$ from membership queries *exactly*, if, for every $n \geq 1$ and $f \in F_n$, $L$ calls $MQ_f$ a number of times and outputs with probability at least $\frac{2}{3}$ a hypothesis $h$ which coincides with $f$, that is $h(x) = f(x)$ for all $x \in \{0,1\}^n$; the maximum number $T(n)$ of oracle calls that $L$ ever makes is the *sample complexity* of $L$.

In the example model [106], the access to $f$ is through an 'example oracle' $EX(f,p)$, where $p$ is a collection of probability distributions $p_n$ over $\{0,1\}^n$, $n \geq 1$; for any value of $n$, when queried, $EX(f,p)$ returns a string $x, f(x)$, with $x \in \{0,1\}^n$ picked randomly under the distribution $p_n$. A classical algorithm $L$ is a *PAC learning algorithm* for $F$, if, for all probability distributions $p$, for all $n \geq 1$, $0 < \epsilon, \delta < 1$, $f \in F_n$, if $L$ is given $n, \epsilon, \delta$ and access to $EX(f,p)$, then with probability at least $1 - \delta$ the algorithm $L$ outputs a hypothesis $h$ for

which the probability under $p$ that $h(x) \neq f(x)$ is at most $\epsilon$. The maximum number $T(n, \epsilon, \delta)$ of calls to $EX(f, p)$ that $L$ ever makes for any $f \in F_n$ and any probability distribution $p_n$ over $\{0,1\}^n$ is the *sample complexity* of $L$.

For the q-q case, Servedio and Gortler define quantum extensions $QMQ_f$ and $QEX(f, p)$ of the classical oracles $MQ_f$ and $EX(f, p)$, and use quantum circuits for the computation [31]; for a lucid introduction to the latter see eg [66]. Roughly, a quantum query for $f \in F$ is an application of a unitary transformation $O_f$ representing $f$, and a quantum learning algorithm $L$ for $F$ is, for every $f \in F$, a composition $L^{(f)}$ of quantum queries $O_f$ intertwined with quantum gates depending on $F$ only, which acts on some initial superposition $\psi_0$, and, upon a measurement of the final superposition $L^{(f)}(\psi_0)$, outputs $f$ with a good probability.

Specifically, the quantum membership query oracle $QMQ_f$ is the quantum black-box oracle for $f$, a quantum gate well-studied in the quantum network model for computing Boolean functions [4, 25, 10]. In this model, a computation is a sequence of unitary transformations ('gates') of a complex vector space $\mathbb{C}^N$, $N = 2^{n+1+m}$, consisting of superpositions of all admissible data $(x, y) \in \{0,1\}^n \times \{0,1\}$ and some working states $z \in \{0,1\}^m$; a designated pair $z', z''$ of working states serves as an output 1 q-bit space $M = \mathbb{C}z' \times \mathbb{C}z''$. For $f \in F_n$, the transformations $O_f$ representing calls to the oracle $QMQ_f$ are of the form $\hat{f} \otimes I_m$, where $\hat{f}$ is defined by the permutation $(x, y) \mapsto (x, y \oplus f(x))$ of data, $\oplus$ denoting addition modulo 2, and $I_m$ is the identity on the working states. A quantum algorithm $L$ for learning $F = \bigcup_n F_n$ *exactly* by *quantum membership queries* is, for every $n \geq 1$, a sequence ('network') of $T = T(n)$ gates $U_0, U_1, \ldots, U_T$ and an initial superposition $\psi_0 \in \mathbb{C}^N$, such that, for all $f \in F_n$, the gate $L_n^{(f)} = U_T O_f U_{T-1} O_f \cdots U_1 O_f U_0$ transforms $\psi_0$ into a superposition $\psi_f$, which, for every $x \in \{0,1\}^n$, upon measurement with respect to the subspace $\mathbb{C}x \otimes M$, yields $(x, f(x))$ with probability at least $\frac{2}{3}$. The function $T(n)$ is the *quantum sample complexity* of $L$.

The quantum example oracle $QEX(f, p)$ in [93] is that of Bshouty and Jackson [23, 24]: a call to $QEX(f, p)$ is a quantum gate $O_f$, as above, but which maps the initial superposition of states to a superposition, in which each state $(x, f(x))$, $x \in \{0,1\}^n$, has the amplitude $\sqrt{p(x)}$. Skipping minor technical details, a *quantum PAC learning algorithm* for $F$ is essentially a family of quantum networks $L$ indexed by $n \geq 1$ and $0 < \epsilon, \delta < 1$, which, for every $f \in F$, making $T(n, \epsilon, \delta)$ calls to $QEX(f, p)$, with probability at least $1 - \delta$ output a function $h$ such that the probability under $p$ that $h(x) \neq f(x)$ is at most $\epsilon$. The *quantum sample complexity* of the PAC algorithm $L$ is the function $T(n, \epsilon, \delta)$.

Servedio and Gortler show in [93] that *quantum and classical learning are polynomially equivalent* in terms of sample complexity. They prove that any class $F$ of Boolean functions, which is quantum exact learnable from quantum membership queries with sample complexity $T(n)$ is also classically exact learnable from membership queries with sample complexity $O(nT^3(n))$. They also prove a corresponding theorem for PAC learning with the bound $O(nT(n))$.

However, these seemingly discouraging results do not make quantum learning uninteresting. Indeed, it is also observed in [93], and further developed by Servedio in [92], that, under some plausible assumptions, quantum learning is *computationally* strictly more efficient than classical learning. The observation in [93] builds on two negative classical learning results of Kearns and Valiant [65] and Angluin and Kharitonov [7], conditional on the hardness of factoring Blum integers. If there is no polynomial-time classical algorithm factoring Blum integers, then by [65] there is a family of Boolean functions which is not efficiently classically PAC learnable, and by [7] there is a family of Boolean functions which is not efficiently classically exact learnable from membership queries. However, by the famous quantum factoring algorithm of Shor [98], both classes are efficiently quantum learnable in the corresponding sense. In [92], this result is strengthened, with the same conclusion for exact learning from membership queries but conditional on the much weaker assumption that *any* one-way function exists, a belief widely held in public-key cryptography.

There are two other papers we found on quantum learning, [90] by Sasaki and Carlini, and [105] by Tucci. It seems however fair to say that the theory of quantum learning, which contrasts classical theory, is today essentially limited to the four papers [18, 24, 92, 93], two of which study the learning of DNF.

## 3 Related work

*Quantum search* It is clear that the notions of learning and search are related, though the exact nature of their relationship is often left to interpretation. In computational context, it is common to see learning as a stronger notion: while search aims at finding an item $x$ satisfying a condition $f(x) = y$, learning aims at determining this condition completely, ie at the computation of $f(x)$ for all $x$ in the domain of $f$. Thus, for example, we search for an instance of a concept, but we learn the concept by finding all its instances.[3]

Quantum search is today a broad field of investigation, which we here only very briefly touch upon; the basics may be found in the expositions [53, 66, 83]. It essentially began in 1996 with Grover's now famous result on quadratic quantum speed-up for unstructured search [47]: while classical search for an item in an unsorted database of $N$ items[4] requires on average no less than $\frac{N}{2}$ queries, for quantum search $O(\sqrt{N})$ queries suffice. For a simple geometric explanation of this algorithm, see Section 8.1 in Kitayev *et al* [66]. Grover's original algorithm has later been refined in various ways, by Grover himself and others, e.g. [20, 19, 49, 48], and analyzed from various angles, e.g. [26, 74, 89].

Grover's algorithm computes the indicator function of the item searched for, in the quantum gate black-box computational model. It was shown earlier (1994) by Bennett *et al* [14] that faster unstructured search is not possible in this model.

[3] Note here the tacit assumption about the 'local nature' of the oracle (eg membership or example); learning turns to search in the presence of a 'global' oracle defined on concepts, for example, telling us whether a candidate concept is the right one or not.

[4] Our distinction between learning and search in this case disappears.

Black-box quantum gate computation of boolean functions has been much investigated, see the survey paper of Buhrman and de Wolf [25]. In particular, it has been shown by Beals *et al* [10] that quantum speed-up in the computation of a *total* boolean function is at most polynomial: 'if a quantum algorithm computes some total boolean function $f$ with bounded error using $N$ black-box queries then there is a classical deterministic algorithm that computes $f$ exactly with $O(N^6)$ queries'; see also Ambainis [4]. Quantifying the distinction between learning and search, this stands in sharp contrast with computation of *partial* boolean functions (when the black box satisfies a particular *promise*): by earlier work of Simon (1994) [99] and Deutch and Jozsa (1992) [32], exponential quantum speed-up is then possible. The role of promise in quantum black-box computation is discussed in the recent paper [21] by Brazier and Plenio.

As a side comment, in view of the decisive role of the oracle in quantum black-box computation and in the related learning models, we note an interesting observation made by Kashevi *et al* [62]: two quantum oracles containing the same classical information may differ exponentially in terms of the size of the register, in the number of calls required to simulate one another.

Though quadratic speed-up of search may in practice be decisive[5], it is not of much help in general search problems, as the size of a search space is exponential in the number of variables, which define it. A standard way of dealing with this situation is to introduce a cost function on the search space and seek near-optimal solutions, trading precision for tractability. The recent work of Trugenberger [104] follows this idea, first appearing in [56], and proposes a 'quantum annealing' probabilistic search heuristic, the accuracy of which depends on a temperature parameter, but not on the size of the search space. See also Hogg [54, 55] for related discussions of quantum combinatorial search. Quantum search for *optimal* solution was investigated by Durr and Hoyer [34], employing Grover's search in an algorithm, and yielding the optimal quadratic speed-up over classical methods.

Quantum algorithms for basic unstructured search, as well as for more general combinatorial search problems, have also been considered in other models of quantum computation. In one such model, proposed by Fahri and Gutman [41], a time-independent Hamiltonian controls the evolution of the quantum computer. In another model, considered in [30, 87], of computation by *adiabatic evolution* [40], the quantum computer evolves according to a time-varying Hamiltonian. A related model, considered by Childs *et al* [27], 'uses only a sequence of measurements to keep the computer near the ground state of a smoothly varying Hamiltonian'.

*Quantum neural networks* Classical neurocomputing has been an important source of inspiration for learning theory; computational learning theory, for example, is naturally packaged with neural network learning [8]. In the words of Ezhov [38], 'the exponential increase of resources ... [the use of exponential

[5] For example, as observed in [19], Grover's algorithm 'cracks the Data Encryption Standard (DES)'.

number of hidden neurons and their interconnections to realize an arbitrary Boolean function] ... turns neurotechnology to reject the goal of precise function realization and faces it with the problem of function approximation using not programming but learning on the restricted set of samples'. Could one expect similar quantum development?

It may be too early to speculate. A relatively recent account of the state of quantum neurocomputing may be found in volume 128 (2000) of *Information Sciences*, which in particular contains papers [3, 11, 36, 79, 103, 117]. We briefly account for other papers we found. Among early papers we find Kak [60] and Chrisley [29], essentially discussing concepts. Ventura *et al* take up varied aspects of prospective quantum neurocomputation in a series of papers [108, 111, 112, 114–116], in particular the question of associative quantum memory; their mathematical techniques largely rely on modifications of Grover's search algorithm. Ezhov [39] considers function approximation using a single quantum neuron, in [37] he looks at pattern recognition, and in [38] he discusses quantum neural processing with respect to interference and entanglement. We also found papers of Morel [75], Shafee [94–96], Andrecut and Ali [5], and Behrman *et al* [12].

We conclude that the subject of quantum neural processing still awaits consolidation in its conceptual, practical, and mathematical treatments. It is not even clear at present what is to be considered a quantum neural network, in the sense of a mathematical model of a physically feasible and computationally adequate computing device. It therefore seems premature to try formulate connections of quantum neurocomputing with learning theory.

*Quantum intelligence, decisions, and games* Put into economic context, a process of learning or search is often a decision process involving actions which affect the learning, and, conversely, most real decision processes involve learning or search. The interplay of learning and acting ('exploration versus exploitation') is classically modelled by Markov decision processes [17, 102] and 'reinforcement learning' [101], and conceptually falls under the heading of 'learning in games' [44]. The learning of decision strategies is treated in less technical language as induction of decision trees [88], and is a standard tool in the machine learning community [73]. Learning problems of this kind have in recent decades attracted considerable interest in theoretical and operational economics [1, 9, 22, 42, 43, 52, 119].

While quantum counterparts of reinforcement learning or learning in games do not seem to have yet been considered, we do now have quantum Markov processes [76], quantum decisions [122], quantum intelligence [61], quantum robots [13], and, most of all, quantum games [35, 67, 63, 70–72, 81, 82]. Practical considerations temporarily aside, there is no shortage here of interesting problems to be put into quantum setting. Besides hoping for quantum computational speedup, one may here trade computational complexity as a measure of hardness of problems, for economic feasibility of their solutions, turning heuristics for hard problems into theorems.

## 4 Prognosis

Any survey of a field of investigation invites speculations as what will happen in the field in the future. We try to refrain from this, but nevertheless note three major trends of development, which quantum models of learning cannot ignore.

First and foremost, and as our account clearly suggests, quantum learning is bound to develop as a side effect of progressing investigations of quantum models of computation and quantum information theory. For example, learning algorithms in the adiabatic evolution model are likely to appear.

Second, it has classically always been the case that pragmatic aspects of computation have had decisive influence on learning problems studied. Learning in games, economic search, genetic algorithms, reinforcement learning, incremental learning, instance-based learning, etc., are likely to find quantum counterparts.

Finally, there is the direction towards mathematical unification of learning models, to encompass both the classical and the quantum cases. This may at present feel rather distant as not even classical learning has yet found proper systematization. In a longer perspective, however, unification is inevitable for communication between the growing number of diverse groups of investigators. We note with interest progresses on logical foundations of quantum mechanics, aiming at a coherent view on both the classical and the quantum theories.

*Note.* In the References below, *arXiv* refers to the site http://xxx.lanl.gov/ and *SiteSeer* refers to the NEC site http://citeseer.nj.nec.com/

## References

1. K. Adam. *Learning while searching for the best alternative.* Working paper ECO 99/4, European University Institute, Department of Economics, Florence, Italy, 1999.
2. R. Ahlswede and I. Wegener. *Search Problems.* Wiley, 1987.
3. E. Alfinito and G. Vitiello. *The dissipative quantum model of brain: how does memory localize in correlated neuronal domains.* Information Sciences 128 (2000), pp. 217-229.
4. A. Ambainis. *Quantum lower bounds by quantum arguments.* ArXiv: quant-ph/0002066
5. M. Andrecut and M. K. Ali. *A Quantum Neural Network Model.* International Journal of Modern Physics C, Vol. 13, no. 1, (2002), pp. 75-88.
6. D. Angluin. *Queries and concept learning.* Machine Learning, vol. 2 (1988), pp. 319-342.
7. D. Angluin and M. Kharitonov. *When won't membership queries help?* J. Comp. Syst. Sci., vol. 50 (1995), pp. 336-355.
8. M. Anthony and P. L. Barlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press, 1999.
9. P. Auer, N. Cesa-Bianchi, Y. Freund and R. E. Schapire. *The nonstochastic multiarmed bandit problem* SIAM Journal on Computing, vol. 32, no. 1 (2002), pp. 48-77.

10. R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. *Quantum lower bounds by polynomials.* Proc. FOCS'98, pp. 351-361. ArXiv: quant-ph/9802049
11. E. C. Behrman, L. R. Nash, J. E. Steck, V. G. Chandrashekar and S. R. Skinner. *Simulations of quantum neural networks.* Information Sciences 128 (2000), pp. 257-269.
12. E. C. Behrman, V. Chandrashekar, Z. Wang, C. K. Belur, J. E. Steck, S. R. Skinner. *A Quantum Neural Network Computes Entanglement.* arXiv: quant-ph/0202131
13. P. Benioff. *Quantum Robots and Environments.* arXiv: quant-ph/9802067
14. C. H. Bennett, E. Bernstein, G. Brassard and U. Vazirani. *Strenghts and weaknesses of quantum computing.* SIAM Journal on Computing, vol. 26, no. 5 (1997), pp. 1510-1524.
15. F. Bergadano and D. Gunetti. *Inductive Logic Programming.* MIT, 1996.
16. A. Berthiaume and G. Brassard. *Oracle quantum computing.* Proc. Workshop on the Physics of Computation, IEEE Press, 1992, pp. 195-199.
17. D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.
18. R. Bonner and R. Freivalds. *Quantum Learning by Finite Automata.* Present collection.
19. M. Boyer, G. Brassard, P. Hoyer and A. Tapp. *Tight bounds on quantum searching.* Fortschritte der Physik, vol. 46 (1998), pp. 493-505.
20. G. Brassard, M. Boyer, M. Mosca and A. Tapp. *Quantum amplitude amplification and estimation.* arXiv: quant-ph/0005055
21. A. Brazier and M. B. Plenio. *Broken Promises and Quantum Algorithms.* arXiv: quant-ph/0304017
22. T. Brenner. *Modelling Learning in Economics.* Edward Elgar Publishers, 1999.
23. N. H. Bshouty and J. C. Jackson. *Learning DNF over the Uniform Distribution using a Quantum Example Oracle.* Proc. COLT (1995), ACM, pp. 118-127.
24. N. H. Bshouty and J. C. Jackson. *Learning DNF Over The Uniform Distribution Using A Quantum Example Oracle.* SIAM Journal on Computing 28 (3) (1999), pp. 1136-1153.
25. H. Buhrman and R. de Wolf. *Complexity Measures and Decision Tree Complexity: A Survey.* 2000. SiteSeer: buhrman00complexity.html
26. M. Butler and P. Hartel. *Reasoning about Grover's quantum search algorithm using probabilistic wp.* ACM Transactions on Programming Languages and Systems, vol. 21, no. 3 (1999), pp. 417-429.
27. A. M. Childs *et al. Quantum search by measurement.* ArXiv: quant-ph/0204013
28. R. J. Chrisley. *Quantum Learning.* In: P. Pylkkänen and P. Pylkkö (eds), New Directions in Cognitive Science, Finnish AI Society, 1995, pp. 77-89.
29. R. J. Chrisley. *Learning in Non-superpositional Quantum Neurocomputers.* In: P. Pylkkänen and P. Pylkkö, (eds), Brain, Mind and Physics, IOS Press, 1997, pp. 126-139.
30. W. van Dam, M. Mosca and U. Vazirani. *How powerful is adiabatic quantum computation?* Proc. FOCS 2001, pp. 279-287.
31. D. Deutsch. *Quantum computational networks.* Proc. Royal Society London, vol. A425 (1989), pp. 73-90.
32. D. Deutsch and R. Jozsa. *Rapid solution of problems by quantum computation.* Proc. Royal Society London, vol. A439 (1992), pp. 553-558.
33. R. M. Dudley. *Uniform Central Limit Theorems.* Cambridge University Press, 1999.
34. C. Durr and P. Hoyer. *A quantum algorithm for finding the minimum.* arXiv: quant-ph/9607014

35. J. Eisert and M. Wilkens. *Quantum games.* arXiv: quant-ph/0004076
36. A. A. Ezhov, A. V. Nifanova, and D. Ventura: *Quantum associative memory with distributed queries.* Information Sciences 128 (2000), pp. 271-293.
37. A. A. Ezhov. *Pattern Recognition with Quantum Neural Networks.* ICAPR 2001, pp. 60-71.
38. A. A. Ezhov. *Role of interference and entanglement in quantum neural processing.* arXiv: quant-ph/0112082
39. A. A. Ezhov, A. G. Khromov, and G. P. Berman. *Multivariable functions approximation using a single quantum neuron.* arXiv: quant-ph/0105134
40. E. Fahri, J. Goldstone, S. Gutman and M. Sipser. *Quantum computation by adiabatic evolution.* arXiv: quant-ph/0001106
41. E. Fahri and S. Gutmann. *Analog analogue of a digital quantum computation.* Phys. Rev. A, vol. 57 (1998), pp. 2403- .
42. T. S. Ferguson. *Optimal Stopping and Applications.* http://www.math.ucla.edu/ tom/Stopping/Contents.html
43. Y. Freund and R. E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of Computer and System Sciences, vol. 55, no. 1, (1997), pp. 119-139.
44. D. Fudenberg and D. K. Levine. *The Theory of Learning in Games.* MIT Press, 1998.
45. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, 1999.
46. E. M. Gold. *Language identification in the limit.* Information and Control, v. 10 (1967), pp. 447-474.
47. L. Grover. *A fast quantum mechanical algorithm for database search.* STOC'96, ACM, pp. 212-219.
48. T. Rudolph and L. Grover. *Quantum searching a classical database.* arXiv: quant-ph/0206066
49. L. Grover. *Tradeoffs in the quantum search algorithm.* arXiv: quant-ph/0201152
50. J. Gruska. *Quantum Computing.* McGraw Hill, 1999.
51. S. Gupta and R. K. P. Zia: *Quantum Neural Networks.* arXiv: quant-ph/0201144
52. D. Haussler. *Decision theoretic generalizations of the PAC model for neural nets and other learning applications.* Information and Computation, vol. 100 (1992), pp. 78-150.
53. M. Hirvensalo. *Quantum Computing.* Springer, 2001.
54. T. Hogg. *Quantum Computing and Phase Transitions in Combinatorial Search.* Journal of Artificial Intelligence Research 4 (1996), pp. 91-128.
55. T. Hogg. *Solving Highly Constrained Search Problems with Quantum Computers.* Journal of Artificial Intelligence Research 10 (1999), pp. 39-66.
56. T. Hogg and D. Portnov. *Quantum Optimization.* Information Sciences 128 (2000), pp. 181-197.
57. J. C. Jackson. *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution.* In: Proc. 35th Symp. on Foundations of Computer Science, 1994, pp. 42-53.
58. J. C. Jackson, J. C., C. Tamon, and T. Yamakami. *Quantum DNF Learnability Revisited.* arXiv: quant-ph/0202066
59. S. Jain, D. Osherson, J. S. Royer and A. Sharma. *Systems that learn, Second edition,* MIT Press, 1999.
60. S. C. Kak. *On Quantum Neural Computing.* Information Sciences 83 (1995), pp. 143-160.

61. S. C. Kak. *Active agents, intelligence and quantum computing.* Information Sciences 128 (2000), pp. 1-17.
62. E. Kashefi, A. Kent, V. Vedral, K. Banaszek. *A Comparison of Quantum Oracles.* arXiv: quant-ph/0109104
63. R. Kay, N. F. Johnson and S. C. Benjamin. *Evolutionary quantum game.* arXiv: quant-ph/0102008
64. M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory.* MIT Press, 1994.
65. M. J. Kearns and L. Valiant. *Cryptographic limitations on learning boolean formulae and finite automata.* Journal of the ACM, vol. 41, no. 1 (1994), pp. 67-95.
66. A. Yu. Kitaev, A. H. Shen and M. N. Vyalyi. *Classical and Quantum Computation.* American Mathematical Society, 2002.
67. C. F. Lee and N. F. Johnson. *Theory of quantum games.* arXiv: quant-ph/0207012
68. M. Li and P. M. B. Vitanyi. *An introduction to Kolmogorov complexity and its applications, 2nd edition.* Springer, 1997.
69. H-K. Lo, S. Popescu, and T. Spiller (Eds.). Introduction to Quantum Computation and Information. World Scientific, 1999.
70. L. Marinatto and T. Weber. *A quantum approach to static games of complete information.* Physics Letters A, vol. 272 (2000), pp. 291-303. Also: arXiv: quant-ph/0004081
71. D. A. Meyer. *Quantum strategies.* Physical Review Letters, vol. 83, no. 5 (1999), pp. 1052-1055.
72. D. A. Meyer. *Quantum games and quantum algorithms.* arXiv: quant-ph/0004092
73. T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
74. A. Miyake and M. Wadati. *Geometric strategy for the optimal quantum search.* arXiv: quant-ph/0109109
75. B. Morel. *Biologically plausible learning rules for neural networks and quantum computing.* Neurocomputing 32-33 (2000), pp. 921-926.
76. P. S. Muhly and B. Solel. *Quantum Markov Processes.* arXiv:math.OA/0203193
77. G. Nakhaeizadeh and C. C. Taylor. *Machine learning and statistics: the interface.* Wiley, 1997.
78. A. Narayanan and M. Moore: *Quantum-inspired genetic algorithms.* International Conference on Evolutionary Computation (1995)
79. A. Narayanan and T. Menneer: *Quantum artificial neural network architectures and components.* Information Sciences 128 (2000), pp. 231-255.
80. B. J. Pearson, J. L. White, T. C. Weinacht, and P. H. Bucksbaum. *Coherent control using adaptive learning algorithms.* arXiv: quant-ph/0008029
81. E. W. Piotrowski. *Quantum market games.* quant-ph/0104006
82. E. W. Piotrowski and J. Sladkowski. *An invitation to quantum game theory.* quant-ph/0211191
83. A. O. Pittenger. *An Introduction to Quantum Computing Algorithms.* Birkhauser, 2001.
84. D. Pollard. *Empirical Processes: Theory and Applications.* NSF-CBMS Regional Conference Series in Probability and Statistics, vol. 2, Institute of Mathematical Statistics, 1990.
85. A. S. Poznyak and K. Najim. *Learning automata and stochastic optimization.* Springer, 1997.
86. C. P. Pritscher. *Quantum Learning: Beyond Duality.* Rodopi, 2001.
87. J. Roland and N. Cerf. *Quantum search by local adiabatic evolution.* arXiv: quant-ph/0107015

88. J. R. Quinlan. *Induction of decision trees.* Machine Learning, vol. 1, no. 1, (1986), pp. 81-106.
89. M. K. Samal and P. Ghose. *Grover's search algorithm and the quantum measurement problem.* arXiv: quant-ph/0202176
90. M. Sasaki and A. Carlini. *Quantum learning and universal quantum matching machine.* arXiv: quant-ph/0202173
91. B. Scholkopf, C. J. C. Burges and A. J. Smola. *Advances in kernel methods: support vector learning.* MIT Press, 1999.
92. R. Servedio. *Separating Quantum and Classical Learning.* In: 28th International Conference on Automata, Languages and Programming (ICALP), 2001, pp. 1065-1080. http://people.deas.harvard.edu/ rocco/papers.html
93. R. Servedio and S. J. Gortler. *Quantum versus Classical Learnability.* In: Proc. 16th IEEE Conference on Computational Complexity (CCC 2001) arXiv: quant-ph/0007036
94. F. Shafee. *Entangled Quantum Networks.* arXiv: quant-ph/0203010
95. F. Shafee. *Neural Networks with c-NOT Gated Nodes.* arXiv: quant-ph/020201
96. F. Shafee. *Semiclassical Neural Network.* arXiv: quant-ph/0202015
97. C. E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, 1949.
98. P. Shor. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM Journal on Computing, vol. 26, no. 5 (1997), pp. 1484-1509.
99. D. Simon. *On the power of quantum computation.* SIAM Journal on Computing, vol. 26, no. 5 (1997), pp. 1474-1483. (Earlier version in FOCS'94).
100. J. Sobel. *Economists' Models of Learning.* Journal of Economic Theory, vol. 94, (2000), pp. 241-261.
101. R. Sutton and A. Barto. *Reinforcement Learning.* MIT Press, 1999.
102. C. Szepesvari and M. L. Littman. *A unified analysis of value-function-based reinforcement-learning algorithms.* Neural Computation, vol. 8, (1999), pp. 217-259.
103. M. Tegmark. *Why the brain is probably not a quantum computer.* Information Sciences 128 (2000), pp. 155-179.
104. C. A. Trugenberger. *Quantum optimization for combinatorial search.* arXiv: quant-ph/0107081
105. R. R. Tucci. *Quantum Computer as an Inference Engine.* arXiv: quant-ph/0004028
106. L. G. Valiant. *A theory of the learnable.* Comm. ACM, vol. 27, no. 11, (1984), pp. 1134-114.
107. V. N. Vapnik. *Statistical learning theory.* Wiley Interscience, 1998.
108. D. Ventura and T. R. Martinez. *An Artificial Neuron with Quantum Mechanical Properties.* Proc. Int. Conf. on Artificial Neural Networks and Genetic Algorithms (1997), pp. 482-485.
109. D. Ventura and T. Martinez. *A Quantum Computational Learning Algorithm.* quant-ph/9807052
110. D. Ventura and T. Martinez. *Quantum Harmonic Sieve: Learning DNF with a Classical Example Oracle.* arXiv: qant-ph/9805043
111. D. Ventura. *Artificial Associative Memory using Quantum Processes.* Proc. Int. Conf. on Computational Intelligence and Neuroscience 2 (1998), pp. 218-221.
112. D. Ventura. *Quantum Associative Memory with Exponential Capacity.* Proc. Int. Joint Conf. on Neural Networks (1998), pp. 509-513.
113. D. Ventura. *Quantum and Evolutionary Approaches to Computational Learning.* Ph.D. Dissertation, Comp. Sc. Dpt., Brigham Young University, 1998.

114. D. Ventura. *Implementing Competitive Learning in a Quantum System.* Proc. Int. Joint Conf. on Neural Networks (1999)
115. D. Ventura and T. R. Martinez. *A Quantum Associative Memory Based on Grover's Algorithm.* Proc. Int. Conf. on Artificial Neural Networks and Genetic Algorithms (1999)
116. D. Ventura and T. R. Martinez. *Quantum associative memory.* Information Sciences 124 (2000), pp. 273-296.
117. D. Ventura and S. C. Kak. *Quantum computing and neural information processing.* Information Sciences 128 (2000), pp. 147-148.
118. M. Vidyasagar. *A Theory of Learning and Generalization.* Springer, 1997.
119. M. Weitzman. *Optimal Search for the Best Alternative.* Econometrica, vol. 47, no. 3, (1979), pp. 641-654.
120. P. Whittle. *Optimization over time.* Wiley, 1982
121. N. Wiener. *Cybernetics, or Control and Communication in the Animal and the Machine.* Second Edition, MIT Press, 1961.
122. M. Zak. *Quantum decision-maker.* Information Sciences 128 (2000), pp. 199-215.

# Learning from Zero Information

Madara Greiziņa and Līga Grundmane

Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia*

**Abstract.** We consider a new type of learning from 'seemingly no information' and introduce a natural complexity measure of this learning. It turns out that probabilistic learning may be of much less complexity.

## 1 Introduction

This paper originates from a problem proposed at the 21st Latvia Open Mathematics Olympiad (1994) [AB98].

> Three persons $A, B$ and $C$ sit around a table. $C$ declares: *I have in mind two subsequent positive integers. One of them is* ... (tells secretly only to $A$), and continues: *The other one is* ... (tells secretly only to $B$).
> After that, a discussion goes on between $A$ and $B$:
> A. *I don't know and I can't know these numbers.*
> B. *I don't know and I can't know these numbers.*
> A. *Now I know these numbers.*
> These three statements were true. Which numbers had $C$ in mind?

Had $C$ told 1 to $A$, his first statement would be false. Hence $C$ told something else to $A$. Hence $B$ has additional information '$C$ has not told 1 to $A$'. Had $C$ told 1 or 2 to $B$, his first statement would be false. Hence $C$ has told something else to $A$. Since the second statement by $A$ was 'Now I know these numbers', the only possibility is '3' and '4'.

This example is a bit surprising since $A$ and $B$ seemingly receive no information. However it was found in cryptography that *zero-knowledge* is a notion to be defined more precisely [Sa90].

## 2 Deterministic learning

In the problem described in our Introduction a specific relation

$$R(x, y) = (\exists z \mid x = z \text{ and } y = z + 1)$$

* Research supported by Grant No.01.0354 from the Latvian Council of Science, Contract IST-1999-11234 (QAIP) from the European Commission, and the Swedish Institute, Project ML-2000

was used. The person $C$ announced that he has guessed two positive integers $x, y$ such that $R(x, y)$. The considered game describes a new kind of complexity, namely, how many repetitions of the phrase 'I don't know and I can't know these numbers' is needed to determine the guess uniquely. We saw in the Introduction that the guess '3' and '4' can be determined by 2 repetitions. It is obvious that $n$ repetitions determine the guess '$n + 1$ and $n + 2$'.

**Theorem 1.** *For arbitrary relation $R(x, y)$ such that $R(x, y)$ well-orders the set of all the positive integers there are infinitely many positive integers $n$ such that there is a $m$ with $R(n, m)$ and to describe this pair of integers one needs no less than $n - 1$ repetitions.*

**Theorem 2.** *For arbitrary relation $R(x, y)$ such that $R(x, y)$ does not well-order the set of all the positive integers there are positive integers $n$ such that there is a positive integer $m$ such that $R(n, m)$ but the pair $(n, m)$ cannot be described uniquely.*

## 3 Probabilistic learning

**Theorem 3.** *Let $R(x, y)$ be an arbitrary relation such that there is a constant $k$ such that for arbitrary positive integer $n$ there are no more than $k$ distinct values of $m$ such that $R(n, m)$. Let $R(x, y)$ partially order the set of all positive integers. Then there are infinitely many positive integers $n$ such that there is a $m$ with $R(n, m)$ and to describe this pair of integers with probability $\frac{1}{k}$ one needs no less than const $\times \log n$ repetitions.*

**Theorem 4.** *For arbitrary positive integer $k$ there is relation $R(x, y)$ such that:*

- *there is a constant $k$ such that for arbitrary positive integer $n$ there are no more than $k$ distinct values of $m$ such that $R(n, m)$,*
- *for all positive integers $n$ such that there is an $m$ with $R(n, m)$, and this pair of integers can be described with probability $\frac{1}{k}$ in const $\times \log n$ repetitions.*

## References

[AAFS01] A. Ambainis, K. Apsītis, R. Freivalds and C. H. Smith. *Hierarchies of probabilistic and team FIN-learning.* Theoretical Computer Science, v. 261, No. 1, 2001, pp. 91-117.

[AB98] A. Andžāns and A. Bērziņš. *Problems and solutions from Latvia Open Mathematics Olympiades.* Zvaigzne Publishers, 1998. (in Latvian)

[BFS00] J. Bārzdiņš, R. Freivalds, and C. H. Smith. *Towards a logic of discovery.* In: R. Bonner and R. Freivalds (Eds.), Quantum Computation and Learning, Proc, Int. Workshop, Sundbyholm, Sweden, 27-29 May 2000, pp. 110-120.

[BFS01] J. Bārzdiņš, R. Freivalds and C. H. Smith. *Towards axiomatic basis of inductive inference.* Lecture Notes in Computer Science 2138, Springer, 2001, pp. 1-13.

[Sa90] A. Salomaa. *Public-Key Cryptography.* Springer-Verlag, 1990.

# Author Index