

Theory of One Tape Linear Time Turing Machines (Extended Abstract)

KOHTARO TADAKI^{**1} TOMOYUKI YAMAKAMI² AND JACK C. H. LIN²

¹ ERATO Quantum Computation and Information Project
Japan Science and Technology Corporation, Tokyo, Japan

² School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada

Abstract. A theory of one-tape linear-time Turing machines is quite different from its polynomial-time counterpart. This paper discusses the computational complexity of one-tape Turing machines of various machine types (deterministic, nondeterministic, reversible, alternating, probabilistic, counting, and quantum Turing machines) that halt in time $O(n)$, where the running time of a machine is defined as the height of its computation tree. We also address a close connection between one-tape linear-time Turing machines and finite state automata.

§1. Model of Computation: Turing Machines. We use a standard definition of an off-line Turing machine. Of special interest is a *one-tape Turing machine* (abbreviated 1TM) $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$, where Q is a finite set of (internal) states, Σ is a nonempty finite input alphabet³, Γ is a finite tape alphabet including Σ , q_0 in Q is an initial state, q_{acc} and q_{rej} in Q are an accepting state and a rejecting state, respectively, and δ is a transition function. Different transition functions δ give rise to various types of 1TMs described in later sections. A *halting state* is either q_{acc} or q_{rej} . Our 1TM is equipped only with one input/work tape such that (i) the tape stretches infinitely to both ends, (ii) the tape is sectioned by cells, and (iii) all cells in the tape are indexed with integers. The tape head starts at the cell indexed 0 (called the *start cell*) and either moves to the right (R) or moves to the left (L), or stays still (N).

In general, a computation of a 1TM forms a tree (called a *computation tree*) whose nodes are configurations. We say that a TM *halts* on input x if *every* computation path of M on input x reaches a certain halting state. An *accepting* (rejecting, resp.) *computation path* is a path terminating in an accepting (rejecting, resp.) state. A 1TM is said to be *synchronized* if all computation paths terminate at the same time on each input.

^{**} Present address: 21st Century COE Program: Research on Security and Reliability in Electronic Society, Chuo University, Tokyo, Japan. This work was partly done while he was visiting the University of Ottawa in November 2001.

³ Throughout this paper, we use the notation Σ to denote an arbitrary nonempty finite alphabet. For a string x over Σ , $|x|$ denotes the *length* of x .

Unlike polynomial-time computation, one-tape linear-time computation is sensitive to the definition of the machine's running time. Such sensitivity is also observed in average-case complexity theory [20]. Michel [15] took the *weak definition* for the running time of a nondeterministic Turing machine (that is, the shortest accepting path) and showed that linear-time nondeterministic 1TMs can recognize non-regular languages. This weak definition therefore gives an enormous power to one-tape nondeterministic machines. From a more realistic point of view, we rather take a *strong definition* (in Michel's term), which defines the running time to be the length of the longest computation path. Throughout this paper, we use the term “*running time*” $\text{Time}_M(x)$ of a 1TM M on input x to mean the height of the computation tree produced by the execution of M on input x ; in other words, the length of the longest computation path of M on x . We often use $T(n)$ to denote a time-bounding function of a given 1TM that maps from the set⁴ \mathbb{N} to \mathbb{N} . Furthermore, a “linear function” means a function of the form $cx + d$ for a certain constant $c, d \in \mathbb{R}^{\geq 0}$ ($= \{r \in \mathbb{R} \mid r \geq 0\}$). A 1TM M is said to run in *linear time* if its running time $\text{Time}_M(x)$ is bounded above by a certain linear function $f(|x|)$, where x is any input.

Although our machine has only one input/work tape, the tape can be split into a constant number of *tracks*. To describe such tracks, we use the following notation. For any pair of symbols $a, b \in \Sigma$, $\begin{bmatrix} a \\ b \end{bmatrix}$ denotes the special tape symbol for which a is written in the upper track and b is written in the lower track of the same cell. For any strings $x, y \in \Sigma^*$ with $|x| = |y|$, $\begin{bmatrix} x \\ y \end{bmatrix}$ denotes the concatenation $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \cdots \begin{bmatrix} x_n \\ y_n \end{bmatrix}$ if $x = x_1x_2 \cdots x_n$ and $y = y_1y_2 \cdots y_n$.

For any 1TM, certain *acceptance criteria* are imposed to define the set of accepted input strings. In general, a language A is said to be *recognized* by a 1TM M of certain machine type if, for every string x , $x \in A$ iff M eventually halts on input x and satisfies the given acceptance criteria.

A *language* (or simply a “set”) *over alphabet* Σ is a subset of Σ^* , and a *complexity class* is a collection of certain languages. The *complement* of A is $\Sigma^* - A$ and often denoted \overline{A} if Σ is clear from the context. For any complexity class \mathcal{C} , the notation $\text{co-}\mathcal{C}$ denotes the *complement* of \mathcal{C} ; that is, the collection of all languages whose complements belong to \mathcal{C} .

The non-regularity plays an important role. For any pair x and y of strings and any integer $n \in \mathbb{N}$, we say that x and y are *n-dissimilar* with respect to a given language L if there exists a string z such that (i) $|xz| \leq n$ and $|yz| \leq n$ and (ii) $xz \in L \iff yz \notin L$. For each $n \in \mathbb{N}$, define $N_L(n)$ (the *non-regularity* of L at n) to be the maximal cardinality of the set in which any distinct pair is *n-dissimilar* with respect to L [6]. It is immediate from the Myhill-Nerode theorem [9] that a language L is regular iff $N_L(n) = O(1)$. This is further improved by the results of Karp [10] and Kaneps and Freivalds [11] as follows: a language L is regular iff $N_L(n) \leq \frac{n}{2} + 1$ for all but finitely-many numbers n in \mathbb{N} .

We assume the reader's familiarity with the notion of *finite (state) automata* (see, e.g., [9]). The class of all *regular* languages is denoted REG. The classes

⁴ Let \mathbb{N} be the set of all natural numbers (i.e., non-negative integers) and set $\mathbb{N}^+ = \mathbb{N} - \{0\}$.

CSL and CFL consist of all *context-sensitive* languages and of all *context-free* languages, respectively.

A *rational (one-head one-way) generalized probabilistic finite automaton* (rational 1GPFA) [17,19] is a 5-tuple $N = (n, \Sigma, \pi, \{M(\sigma) \mid \sigma \in \Sigma\}, F)$, where (i) $n \in \mathbb{N}^+$ is the number of states, (ii) Σ is an alphabet, (iii) π is a row vector which has n rational components, (iv) for each $\sigma \in \Sigma$, $M(\sigma)$ is an $n \times n$ matrix whose elements are rational numbers, and (v) $F \subseteq \{1, \dots, n\}$ is the set of accepting states. A *word matrix* $M(x)$ ($x \in \Sigma^*$) of N is defined as $M(\lambda) = I_n$ for the *empty string* λ , where I_n is the identity matrix of order n and $M(x_1 \dots x_k) = M(x_1) \dots M(x_k)$ for $x_1, \dots, x_k \in \Sigma$. For each $x \in \Sigma^*$, the *acceptance function* $p_N(x)$ is defined as $\pi M(x) \eta^F$, where η^F is the column vector whose i -th component is equal to 1 or 0 depending on whether $i \in F$ or not. By 1GAF_{rat} , we denote the set of all acceptance functions of rational 1GPFAs. For each $\varepsilon \in [0, 1]$, let $T(N, \varepsilon) = \{x \in \Sigma^* \mid p_N(x) > \varepsilon\}$ and $T^=(N, \varepsilon) = \{x \in \Sigma^* \mid p_N(x) = \varepsilon\}$. Moreover, a *rational (one-head one-way) probabilistic finite automaton* (rational 1PFA) [16] is a rational 1GPFA $(n, \Sigma, \pi, \{M(\sigma) \mid \sigma \in \Sigma\}, F)$ such that (i) all components of π are in $[0, 1]$ and their sum is 1, and (ii) for each $\sigma \in \Sigma$, all elements of $M(\sigma)$ are in $[0, 1]$ and the sum of all the elements of each row of $M(\sigma)$ is 1. For any rational 1PFA N , since $p_N(x)$ equals the probability that N accepts x , $p_N(x)$ is called the *acceptance probability* of N on input x . Let GSL_{rat} and SL_{rat} be the collections of all sets $T(N, 1/2)$ for certain rational 1GPFAs N and for certain rational 1PFAs, respectively. Similarly, $\text{GSL}_{\text{rat}}^=$ and $\text{SL}_{\text{rat}}^=$ are defined by substituting $T^=(N, 1/2)$ for $T(N, 1/2)$. Turakainen [19] showed that the equivalence of GSL_{rat} and SL_{rat} . With a similar idea, we can show that $\text{GSL}_{\text{rat}}^= = \text{SL}_{\text{rat}}^=$.

Finally, all logarithms in this paper are to the base two.

§2. Deterministic and Reversible Computation. This section establishes a basic collapse result of linear-time deterministic reversible 1TMs. Since the notation DLIN is widely used for the model of multiple-tape linear-time Turing machines, we use the following new notations to emphasize our one-tape model. The notation $1\text{-DTime}(T(n))$ denotes the collection of all languages recognized by $T(n)$ -time deterministic⁵ 1TMs. Given a set \mathcal{T} of time-bounding functions, $1\text{-DTime}(\mathcal{T})$ is the union of $1\text{-DTime}(T(n))$ for every function T in \mathcal{T} . The *one-tape deterministic linear-time* complexity class 1-DLIN is then defined to be $1\text{-DTime}(O(n))$.

Earlier, Hennie [8] proved that $\text{REG} = 1\text{-DLIN}$. Elaborating Hennie's argument, Kobayashi [12] improved his result by showing $\text{REG} = 1\text{-DTime}(o(n \log n))$. This bound $o(n \log n)$ is optimal because $1\text{-DTime}(O(n \log n))$ contains non-regular languages, e.g., $\{a^n b^n \mid n \in \mathbb{N}\}$ and $\{a^{2^n} \mid n \in \mathbb{N}\}$.

Proposition 1. [8,12] $\text{REG} = 1\text{-DTime}(o(n \log n)) \neq 1\text{-DTime}(O(n \log n))$.

For later use, we define the functional version of 1-DLIN. A function from Σ^* to Σ^* is in 1-FLIN if there exists a deterministic 1TM M satisfying that,

⁵ Its transition function δ is a map from $(Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma$ to $Q \times \Gamma \times \{L, N, R\}$.

on any input x , (i) M halts by entering *any* halting state in time linear in $|x|$ and (ii) when M halts, the tape consists only of $f(x)$ (surrounded by the blank symbols) with the leftmost symbol of $f(x)$ written in the start cell.

Damm and Holzer [5] considered Karp-Lipton type advice for automata. To make most of advice, we take a slightly different formulation for 1TMs. In general, for any complexity class \mathcal{C} based on 1TMs, the notation \mathcal{C}/n represents the collection of all languages A such that there exist an alphabet Σ , a set B in \mathcal{C} , and a length-preserving⁶ function h from \mathbb{N} to Σ^* (called an *advice function*) such that, for every $x \in \Sigma^*$, $x \in A$ iff $\left[h\left(\frac{x}{|x|}\right) \right] \in B$. For instance, any language A whose density $|A \cap \Sigma^n|$ is always at most 1 clearly belongs to REG/n . However, the set $L_{\text{num}} = \{x \in \{0, 1\}^* \mid \#_0(x) = \#_1(x)\}$, where $\#_i(x)$ denotes the number of symbol i in x , does not belong to REG/n .

Lemma 1. *The language L_{num} is not in REG/n .*

Proof. Let $\Sigma = \{0, 1\}$. Assuming that $L_{\text{num}} \in \text{REG}/n$, choose a deterministic finite automaton $M = (Q, \Sigma, q_0, F)$ and an advice function h from \mathbb{N} to Σ^* such that, for every $x \in \Sigma^*$, $x \in L_{\text{num}}$ iff $tr_h(x) \in B$, where $tr_h(x)$ denotes $\left[h\left(\frac{x}{|x|}\right) \right]$. Take the minimal integer n satisfying $n + 1 > |Q|$. For each $k \in \{0, 1, \dots, n\}$, define y_k to be any string of length n satisfying $\#_0(y_k) = k$. Hereafter, we focus on strings yz 's of length $2n$.

There exist two distinct $k, l \in \{0, 1, \dots, n\}$ such that (i) $y_k z_k, y_l z_l \in L_{\text{num}}$ for some $z_k, z_l \in \Sigma^n$ and (ii) M enters the same internal state after reading y_k as well as y_l (since $n + 1 > |Q|$). It follows from these conditions that M also accepts input $tr_h(y_k z_l)$. However, we have $\#_0(y_k z_l) \neq \#_1(y_k z_l)$, a contradiction. Therefore, $L_{\text{num}} \notin \text{REG}/n$. \square

In the early 1970s, Bennett [2] initiated a study of reversible computation. We take the definition given in [3] in connection to quantum Turing machines in §7. A *reversible 1TM* is a deterministic 1TM of which each configuration has at most one predecessor configuration. We use the notation $1\text{-revDTime}(T(n))$ to denote the collection of all languages recognized by $T(n)$ -time reversible 1TMs and let $1\text{-revDTime}(\mathcal{T})$ be $\bigcup_{T \in \mathcal{T}} 1\text{-revDTime}(T(n))$. Finally, let $1\text{-revDLIN} = 1\text{-revDTime}(O(n))$. Obviously, $1\text{-revDLIN} \subseteq 1\text{-DLIN}$.

Kondacs and Watrous [13] recently demonstrated that any one-head one-way deterministic finite automaton can be simulated in linear time by a certain on-head two-way reversible finite automaton. Since any one-head two-way reversible finite automaton is indeed a reversible 1TM, we obtain that $\text{REG} \subseteq 1\text{-revDLIN}$. Proposition 1 thus concludes:

Proposition 2. $\text{REG} = 1\text{-revDLIN} = 1\text{-revDTime}(o(n \log n))$.

§3. Nondeterministic Computation. Nondeterminism has been widely studied in the literature since many problems naturally arising in computer science have nondeterministic traits.

⁶ A function from \mathbb{N} to Σ^* is called *length-preserving* if $|f(n)| = n$ for all $n \in \mathbb{N}$.

Similar to the deterministic case, let $1\text{-NTime}(T(n))$ denote the collection of all languages recognized by $T(n)$ -time nondeterministic⁷ 1TMs and let $1\text{-NTime}(\mathcal{T})$ be the union of all $1\text{-NTime}(T(n))$ for all $T \in \mathcal{T}$. We define the *one-tape nondeterministic linear-time* class 1-NLIN to be $1\text{-NTime}(O(n))$.

We expand Kobayashi's collapse result on $1\text{-DTime}(o(n \log n))$ and show the following theorem.

Theorem 1. $\text{REG} = 1\text{-NTime}(o(n \log n)) \neq 1\text{-NTime}(O(n \log n))$.

The proof of Theorem 1 consists of two technical lemmas: Lemmas 2 and 3. The first lemma is a core of Kobayashi's argument in [12, Theorem 3.3] and the second lemma is due to Hennie [8, Theorem 2].

Now, we introduce the key terminology for the lemmas. Let M be any 1TM. Any boundary that separates two adjacent cells in M 's tape is called an *intercell boundary*. The *crossing sequence* at intercell boundary b along computation path s of M is the sequence of inner states of M at the time when the tape head crosses b , first from left to right, and then alternately in both directions. Assume that an input/work tape contains a string x , which may be surrounded by non-blank symbols. The *right-boundary* of x is the intercell boundary between the rightmost symbol of x and its right-adjacent symbol. Similarly, the *left-boundary* of x is defined as the intercell boundary between the leftmost symbol of x and its left-adjacent symbol. Any intercell boundary between the right-boundary and the left-boundary of x (including both ends) is called a *critical-boundary* of x .

Lemma 2 comes from the close observation of Kobayashi's argument that (i) the acceptance criteria of nondeterministic 1TMs are irrelevant and (ii) the argument is applicable to any other models of 1TMs dealt with in this paper.

Lemma 2. *Assume that $T(n) = o(n \log n)$. For any $T(n)$ -time nondeterministic 1TM M , there exists a constant $c \in \mathbb{N}$ such that, for each string x , any crossing sequence at every intercell boundary in every computation path of M on input x has length at most c . In addition, no acceptance criteria of the machine are necessary.*

In essence, Hennie [8] proved that any deterministic computation with short crossing sequences has non-regularity bounded above by a certain constant. We generalize his result to the nondeterministic case in the following lemma. Different from the previous lemma, Lemma 3 relies on the acceptance criteria of nondeterministic 1TMs. Nonetheless, Lemma 3 does not refer to rejecting computation paths.

Lemma 3. *Let L be any language and let M be any nondeterministic 1TM that recognizes L . For each $n \in \mathbb{N}$, let S_n be the set of all crossing sequences at any critical-boundary along any accepting computation path of M on any input of length $\leq n$. Then, $N_L(n) \leq 2^{|S_n|}$ for all $n \in \mathbb{N}$, where $|S_n|$ denotes the cardinality of S_n .*

⁷ Its transition function δ is a map from $(Q - \{q_{acc}, q_{rej}\}) \times \Gamma$ to $2^{Q \times \Gamma \times \{L, N, R\}}$, where 2^A denotes the power set of A .

Since REG is closed under complementation, so is $1\text{-NTime}(o(n \log n))$. In contrast, a simple crossing sequence argument proves that $1\text{-NTime}(O(n \log n))$ does not contain the set of palindromes $L_{pal} = \{x \in \{0, 1\}^* \mid x = x^R\}$, where x^R is the reverse of x . Since $\overline{L_{pal}} \in 1\text{-NTime}(O(n \log n))$, $1\text{-NTime}(O(n \log n))$ is different from $\text{co-}1\text{-NTime}(O(n \log n))$.

Corollary 1. *The class $1\text{-NTime}(o(n \log n))$ is closed under complementation whereas $1\text{-NTime}(O(n \log n))$ is not closed under complementation.*

The reducibility has played a central role in the theory of NP-completeness. Similarly, we can introduce the following restricted reducibility. Let \mathcal{T} be any set of time-bounding functions. A set A over alphabet Σ_1 is *many-one $1\text{-NTime}(\mathcal{T})$ -reducible* to another set B over alphabet Σ_2 (notationally, $A \leq_m^{1\text{-NTime}(\mathcal{T})} B$) if there exist a function $T \in \mathcal{T}$ and a nondeterministic 1TM M such that, for every string x in Σ_1^* , (i) M on input x halts within time $T(|x|)$ with the tape consisting only of one block of non-blank symbols, say y_p , on each computation path p , provided that the first symbol of y_p must be in the start cell, (ii) when M halts, the tape head returns to the start cell along each computation path, and (iii) $x \in A$ iff $y_p \in B$ for a certain computation path p of M on input x . We use the notation $1\text{-NTime}(\mathcal{T})_m^B$ to denote the collection of all languages A that are many-one $1\text{-NTime}(\mathcal{T})$ -reducible to B .

A straightforward simulation shows that $1\text{-NTime}(o(n \log n))_m^{\text{REG}}$ is included in $1\text{-NTime}(o(n \log n))$. Since $\text{REG} = 1\text{-NLIN}$, 1-NLIN is closed under many-one $1\text{-NTime}(o(n \log n))$ reductions. This result will be used in §4.

Proposition 3. *The class 1-NLIN is closed under many-one $1\text{-NTime}(o(n \log n))$ -reductions.*

As a special case, we say that A is *many-one 1-NLIN -reducible* to B if A is many-one $1\text{-NTime}(O(n))$ -reducible to B and we then define the relativized complexity class 1-NLIN_m^B . Similarly, we can define the “many-one 1-DLIN -reducibility” and its corresponding relativized class 1-DLIN_m^B . The oracle separation is possible between 1-DLIN_m^B and 1-NLIN_m^B .

Proposition 4. *There exists an oracle B such that $1\text{-DLIN}_m^B \neq 1\text{-NLIN}_m^B$.*

§4. Alternating Computation. Chandra, Kozen, and Stockmeyer introduced the concept of alternating Turing machines as a natural extension of nondeterministic Turing machines. An *alternating 1TM* is similar to a nondeterministic 1TM except that its internal states are labeled⁸ either \exists (existential) or \forall (universal). The *k-alternation* means that the number of the times when states change between \exists and \forall equals k . Let k and T be any functions from \mathbb{N} to \mathbb{N} . The notation $1\text{-}\Sigma_{k(n)}\text{Time}(T(n))$ denotes the collection of all languages recognized by certain $T(n)$ -time alternating 1TMs with at most $k(n)$ -alternation starting

⁸ This labeling is done by the function g that maps Q to $\{\exists, \forall\}$.

with an \exists -state. In particular, we write $1-\Sigma_{k(n)}^{\text{LIN}}$ for $1-\Sigma_{k(n)}\text{Time}(O(n))$. In contrast, let 1-ALIN be the collection of all languages recognized by linear-time alternating 1TMs with unbounded alternation.

Here, we consider the case of constant alternation. Clearly, $\text{REG} \subseteq 1-\Sigma_k^{\text{LIN}} \subseteq 1\text{-ALIN}$ for any $k \in \mathbb{N}^+$. In what follows, we prove that $1-\Sigma_k^{\text{LIN}}$ collapses to REG.

Theorem 2. $\text{REG} = \bigcup_{k \in \mathbb{N}^+} 1-\Sigma_k^{\text{LIN}}$.

The proof of Theorem 2 proceeds by induction on k . The base case $k = 1$ is already shown in Theorem 1 since an alternating 1TM with 1-alternation starting with an \exists -state is identical to a nondeterministic 1TM. The induction step $k > 1$ follows from Proposition 3 and Lemma 4. In Lemma 4, we show that alternation can be viewed as an application of many-one 1-NLIN-reductions.

Lemma 4. *Let $k \in \mathbb{N}$. A set A is in $1-\Sigma_{k+1}^{\text{LIN}}$ iff A is many-one 1-NLIN-reducible to \overline{B} for a certain set B in $1-\Sigma_k^{\text{LIN}}$.*

Proof. (Only If – part) Take any linear-time alternating 1TM M with at most k -alternation that recognizes a given set A over alphabet Σ_1 . Without loss of generality, we can assume that M never visits the cell indexed -1 since, otherwise, we can “fold” a computation into two tracks, in which the first track simulates the tape region of nonnegative indices, and the second track simulates the tape region of negative indices.

First, we define the linear-time nondeterministic 1TM M' as follows. On input x , M' marks the start cell and then simulates M . Whenever M writes a blank symbol, replace it with a new symbol not in Σ_1 , say $\#$. If M enters the first \forall -state p , M' marks the cell (by changing its tape symbol a to the new symbol $\begin{bmatrix} a \\ p \end{bmatrix}$), moves the head back to the start cell, and erases the mark. It is important to note that M' has at least $|x|$ non-blank symbols in its tape when it halts. Let Σ consist of all symbols of the form $\begin{bmatrix} a \\ p \end{bmatrix}$ for any symbol $a \in \Sigma_1$ and any \forall -state p .

Next, we define N as follows. Let $\Sigma_2 = \Sigma_1 \cup \Sigma \cup \{\#\}$. On input y in Σ_2^* , N changes all $\#$ s to the blank symbol, finds in the tape the first symbol from Σ , say $\begin{bmatrix} a \\ p \end{bmatrix}$, and changes it back to a . Recover the state p as well. By starting with this \forall -state p , N simulates M . The desired set B consists of all input strings rejected by N . Obviously, B is in $1-\Sigma_{k-1}^{\text{LIN}}$. It is also not difficult to show that A is many-one 1-NLIN-reducible to \overline{B} via M' .

(If – part) The proof is done by induction on $k \in \mathbb{N}$. Assume that A is many-one 1-NLIN-reducible to \overline{B} via a reduction machine N , where B is in $1-\Sigma_k^{\text{LIN}}$. Choose a linear-time alternating 1TM M that recognizes B with k -alternation starting with an \exists -state. Define \overline{M} to be the one obtained from M by exchanging \forall -states and \exists -states and swapping an accepting state and a rejecting state. Now, we define N' as follows: on input x , run N , and, when it halts, run \overline{M} . Clearly, N' runs in linear time. It is also easy to show that N' recognizes A with $(k+1)$ -

alternation. Thus, A belongs to $1-\Sigma_{k+1}^{\text{LIN}}$. \square

§5. Probabilistic Computation. Probabilistic (or randomized) computation has been proven to be essential to many applications in computer science. Probabilistic extensions of deterministic Turing machines have been studied since as early as the 1950s. For simplicity, we use Gill's notion of probabilistic Turing machines with *fair coin* flips. Formally, a *probabilistic 1TM* is defined as a non-deterministic 1TM that has *at most* two nondeterministic choices at each step, which is often called a *coin toss* (or *coin flip*) if there are exactly two choices. Each coin toss is made with probability $1/2$. Instead of taking an expected running time, we define a probabilistic 1TM M to be $T(n)$ -time bounded if, for every x , any computation path of M on input x has length at most $T(|x|)$. The probability associated with computation path s equals 2^{-m} , where m is the number of coin tosses along path s . The *acceptance probability* of M on input x , denoted $p_M(x)$, is the sum of all probabilities of any accepting computation paths. We say that M *recognizes L with error probability at most ε* if, for every x , (i) if $x \in L$, then $1 - p_M(x) \leq \varepsilon$; and (ii) if $x \notin L$, then $p_M(x) \leq \varepsilon$.

We begin with a key lemma, which is a probabilistic version of Lemma 3. Kaneps and Freivalds [11], following Rabin's [16] result, proved a similar result for probabilistic finite automata.

Lemma 5. *Let L be any language and let M be any probabilistic 1TM that recognizes L with error probability at most $\varepsilon(n)$, where $0 \leq \varepsilon(n) < 1/2$ for all $n \in \mathbb{N}$. For each $n \in \mathbb{N}$, let S_n be the set of all crossing sequences at any critical-boundary of x along any accepting computation path of M on every input x of length $\leq n$. Then, $N_L(n) \leq 2^{\lceil |S_n| \lceil |S_n| / \delta(n) \rceil \rceil}$ for all $n \in \mathbb{N}$, where $\delta(n) = 1/2 - \varepsilon(n)$.*

Proof. Fix $n \in \mathbb{N}$ arbitrarily. For every string $x \in \Sigma^{\leq n}$ and every crossing sequence $v \in S_n$, let $w_l(x|v)$ ($w_r(v|z)$, resp.) be the sum, over all z (all x , resp.) with $|xz| \leq n$, of all probabilities of the coin tosses made during the head staying in the left-side region of the right-boundary of x (the right-side region of the left-boundary of z , resp.) along any accepting computation path of M on input xz . By their definitions, we have $0 \leq w_l(x|v), w_r(v|z) \leq 1$. The key observation is that the acceptance probability of M on input xz with $|xz| \leq n$ equals $\sum_{v \in S_n} w_l(x|v)w_r(v|z)$.

Now, we say that x *n -supports* (i, v) if $|x| \leq n$, $i \in \{0, 1, \dots, \lceil |S_n| / \delta(n) \rceil - 1\}$, $v \in S_n$, and $i \cdot \delta(n) / |S_n| \leq w_l(x|v) \leq (i+1)\delta(n) / |S_n|$. Define the set $\text{Supp}_n(x) = \{(i, v) \mid x \text{ } n\text{-supports } (i, v)\}$. We first show that, for every x, y, z with $|xz| \leq n$ and $|yz| \leq n$, if $xz \in L$ and $\text{Supp}_n(x) = \text{Supp}_n(y)$, then $yz \in L$. This is shown as follows. Since $\text{Supp}_n(x) = \text{Supp}_n(y)$, $|w_l(x|v) - w_l(y|v)| \leq \delta(n) / |S_n|$ for all $v \in S_n$. Thus, $|p_M(xz) - p_M(yz)| = |\sum_{v \in S_n} (w_l(x|v) - w_l(y|v)) \cdot w_r(v|z)| \leq \sum_{v \in S_n} |w_l(x|v) - w_l(y|v)| \leq \sum_{v \in S_n} \delta(n) / |S_n| = \delta(n)$. Since $xz \in L$, we obtain $p_M(xz) \geq 1 - \varepsilon(n)$, which yields $p_M(yz) > \varepsilon(n)$. Therefore, $yz \in L$.

Note that $N_L(n)$ is bounded above by the number of distinct $\text{Supp}_n(x)$'s for all $x \in \Sigma^{\leq n}$. Therefore, $N_L(n)$ is at most $2^{\lceil |S_n| \rceil \lceil |S_n|/\delta(n) \rceil}$. \square

For any language L , we say that M *recognizes L with bounded-error probability* if there exists a constant $\varepsilon > 0$ such that M recognizes L with error probability at most $\frac{1}{2} - \varepsilon$. We define $1\text{-BPTIME}(T(n))$ as the collection of all languages recognized by $T(n)$ -time probabilistic 1TM with bounded error probability. We also define $1\text{-BPTIME}(\mathcal{T})$ similar to the nondeterministic case. The *one-tape bounded-error probabilistic linear-time* class 1-BPLIN is $1\text{-BPTIME}(O(n))$.

Let L be any language recognized by a probabilistic 1TM M with bounded-error probability in time $o(n \log n)$. Lemmas 2 and 5 guarantee that $N_L(n) = O(1)$. Therefore, we obtain the following theorem.

Theorem 3. $\text{REG} = 1\text{-BPTIME}(o(n \log n)) \neq 1\text{-BPTIME}(O(n \log n))$.

Next, we consider unbounded-error probabilistic computation. We define 1-PLIN to be the collection of all languages of the form $\{x \in \Sigma^* \mid p_M(x) > 1/2\}$ for certain linear-time probabilistic 1TMs M . It is easy to show that 1-PLIN is closed under complementation and symmetric difference⁹.

The following theorem establishes an automaton-characterization of 1-PLIN . We write 1-synPLIN for the subset of 1-PLIN defined by linear-time probabilistic 1TMs that are particularly *synchronized*.

Theorem 4. $1\text{-PLIN} = 1\text{-synPLIN} = \text{SL}_{\text{rat}}$.

Theorem 4 follows from Lemmas 6 and 7. We begin with Lemma 6.

Lemma 6. $\text{SL}_{\text{rat}} \subseteq 1\text{-synPLIN}$.

Proof. Given a rational 1PFA N , assume that the set of all transition probabilities of N is of the form $\{r_1/m, r_2/m, \dots, r_k/m\}$ for certain positive integers m, r_1, r_2, \dots, r_k . A new probabilistic 1TM M simulates each step of N by first generating $2^{\lceil \log m \rceil}$ branches without moving its head. The first $2^{\lceil \log m \rceil} - m$ branches are referred to as *marked* and the rest is used for simulating N 's step. When N finishes scanning the entire input, M makes one more coin flip and enters N 's last states (in case where they are not final states, we force M to enter q_{rej}) except that, if M 's computation path includes at least one marked branch, M accepts and rejects with the equal probability. This simulation makes M 's computation paths not only terminate all at once but also toss the equal number of coins. \square

The following lemma complements Lemma 6 since $1\text{-synPLIN} \subseteq 1\text{-PLIN}$.

Lemma 7. $1\text{-PLIN} \subseteq \text{SL}_{\text{rat}}$.

Proof. Assume that L is any set in 1-PLIN . First, we build a linear-time probabilistic 1TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ with the following five conditions:

⁹ The *symmetric difference* between two sets A and B is $(A - B) \cup (B - A)$.

(i) M recognizes L with unbounded-error probability, (ii) M never visits the cell indexed -1, (iii) M modifies tape symbols only in the *input area* (i.e., the block of cells that initially consists of an input string), (iv) M does not enter q_0 from any other states, and (v) M finally crosses the right-boundary of input x by entering a halting state. For simplicity, we can assume that, when the input is the empty string λ , M immediately enters a halting state without moving its head. The construction of such a restrictive machine is done by first marking the right and left end of the input area and then by “folding” the computation into a constant number of tracks of the tape. To help the later description, we assume that M first moves its head to the right to find the first blank symbol in state q_1 and steps back to mark the right-end marker by entering state q_2 . After this marking process, M ’s head never scans the right-side region of the right-end marker except for the final step (at which the machine enters a halting state). Thus, the crossing sequence at the right-boundary of x can be assumed to be (q_1, q_2, q) , where $q \in \{q_{acc}, q_{rej}\}$.

The goal is to construct the rational 1GPFA N for which $L = T(N, 1/2)$. Let S be the set of all crossing sequences of M . For convenience, we assume that S includes the sequence (q_0) . By Lemma 2, all such crossing sequences have their length bounded above by a certain constant. Let σ be any symbol in Σ . For any pair (u, v) of elements in S , we define $P(u; \sigma; v)$ as the *probability* of the following event: for a certain pair of strings y and z , the following holds. Consider the computation where M starts on input $y\sigma z$ with the head initially scanning σ in state q , which is the first entry of u . In this computation, (i) if $u \neq (q_0)$, then u coincides with the crossing sequence at the left-boundary of σ , (ii) if $u = (q_0)$, then M never crosses the left-boundary of σ , and (iii) v coincides with the crossing sequence at the right-boundary of σ .

Obviously, $P(u; \sigma; v)$ is a rational number. It follows that, for each $x = \sigma_1 \cdots \sigma_m \in \Sigma^*$ where $m \geq 1$ and each σ_i is in Σ , $p_M(x) = \sum_{\mathbf{v}} \prod_{i=1}^m P(v_{i-1}; \sigma_i; v_i)$, where the summation is taken over all sequences $\mathbf{v} = (v_1, \dots, v_{m-1})$ of S with $v_0 = (q_0)$ and $v_m = (q_1, q_2, q_{acc})$.

The desired rational 1GPFA $N = (|S|, \Sigma, \pi, \{M'(\sigma) \mid \sigma \in \Sigma\}, F)$ is defined as follows. Let f be any bijection from S to $\{1, \dots, |S|\}$. For each $\sigma \in \Sigma$, let $M'(\sigma)$ be the $|S| \times |S|$ matrix whose (i, j) -element is $P(f^{-1}(i); \sigma; f^{-1}(j))$ for any $i, j \in \{1, \dots, |S|\}$. Let π be the row vector whose i -th component is equal to 1 (0, resp.) if $i = f(q_0)$ ($i \neq f(q_0)$, resp.). We set $F = \{f(q_0)\}$ if $\lambda \in L$ and $F = \{f(q_1 q_2 q_{acc})\}$ otherwise. It is easy to verify that, for every input $x \neq \lambda$, $p_N(x) = \pi M'(x) \eta^F = p_M(x)$. Hence, we have $L = T(N, 1/2)$; and therefore, $L \in \text{GSL}_{rat} = \text{SL}_{rat}$. \square

Dwork and Stockmeyer [7] demonstrated that the set $L_{cent} = \{x1y \mid x, y \in \{0, 1\}^*, |x| = |y|\}$ is in $\text{CFL} - \text{SL}_{rat}$. Note that L_{cent} is also in REG/n . Moreover, Macarie [14] showed the proper containment $\text{SL}_{rat} \subsetneq L$, where L is the class of all languages recognized by multiple-tape deterministic Turing machines, with a read-only input tape and read/write work-tapes, that uses $O(\log n)$ tape-space in all tapes except for the input tape. We thus conclude the following corollary of Theorem 4.

Corollary 2. $\text{CFL} \cap \text{REG}/n \not\subseteq 1\text{-PLIN}$ and $1\text{-PLIN} \subsetneq \text{L}$.

§6. Counting Computation. Any decision problem of asking whether there exists a “solution” naturally induces the problem of counting the number of such solutions. A *counting 1TM* is a variant of a nondeterministic 1TM, which behaves like a nondeterministic 1TM except that, when it halts, it outputs the number of its accepting computation paths. Let $\#M(x)$ denote the outcome of a counting 1TM M on input x .

Similar to Valiant’s $\#P$, we use the notation $1\text{-}\#\text{LIN}$ (pronounced “one sharp lin”) for the collection of all functions f , from Σ^* to \mathbb{N} , which are computed by certain linear-time counting 1TMs. This class $1\text{-}\#\text{LIN}$ naturally includes 1-FLIN by identifying any natural number n with the n th string over alphabet Σ (in the standard order). Another useful function class is 1-GapLIN , which is defined as the class of all functions whose values are the difference between the number of accepting paths and the number of rejecting paths of linear-time nondeterministic 1TMs. Such functions are conventionally called *gap functions*. It is easy to show the following closure properties: for any functions f and g in 1-GapLIN , the functions $f \cdot g$, $f + g$, and $f - g$ all belong to 1-GapLIN . Moreover, 1-GapLIN is a proper subset of 1GAF_{rat} .

Lemma 8. $1\text{-GapLIN} \subsetneq 1\text{GAF}_{\text{rat}}$.

Proof. The lemma is proven in two steps. Firstly, the proof technique of Lemma 7 allows us to prove the containment $1\text{-}\#\text{LIN} \subseteq 1\text{GAF}_{\text{rat}}$ by defining $P(u; \sigma; v)$ to be the number of computation paths instead of probabilities. Secondly, since $1\text{-GapLIN} \subseteq \{f - g \mid f, g \in 1\text{-}\#\text{LIN}\}$, the lemma is immediate from the following closure property of 1GAF_{rat} : for any $f_1, f_2 \in 1\text{GAF}_{\text{rat}}$ and any $c_1, c_2 \in \mathbb{Q}$, $c_1 f_1 + c_2 f_2$ is in 1GAF_{rat} . The inequality is trivial. \square

Lemmas 6 and 8 build a bridge between counting computation and unbounded-error probabilistic computation.

Proposition 5. $1\text{-PLIN} = \{A \mid \exists f \in 1\text{-GapLIN} [A = \{x \mid f(x) > 0\}]\}$.

We introduce the counting class 1-SPLIN as the collection of all languages whose characteristic functions¹⁰ belong to 1-GapLIN . Moreover, the class $1\text{-}\oplus\text{LIN}$ (pronounced “one parity lin”) consists of all languages of the form $\{x \in \Sigma^* \mid f(x) \equiv 1 \pmod{2}\}$ for certain functions f in $1\text{-}\#\text{LIN}$. It follows that $\text{REG} \subseteq 1\text{-SPLIN} \subseteq 1\text{-}\oplus\text{LIN}$. We prove that these classes collapse to REG .

Theorem 5. $\text{REG} = 1\text{-SPLIN} = 1\text{-}\oplus\text{LIN}$.

Proof. It suffices to show that $1\text{-}\oplus\text{LIN} \subseteq \text{REG}$. This can be shown by modifying the proof of Lemma 5. Here, we define $w_l(x|v)$ and $w_r(v|z)$ to denote the number of accepting computation paths instead of probabilities. We also

¹⁰ The *characteristic function* χ_A of a set A is defined as $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ otherwise.

define $\text{Supp}_n(x)$ to be the set $\{v \in S_n \mid w_l(x|v) \equiv 0 \pmod{2}\}$. For any x, y with $|xz| \leq n$ and $|yz| \leq n$, we can prove that, if $xz \in L$ and $\text{Supp}_n(x) = \text{Supp}_n(y)$, then $\#M(xz) - \#M(yz) \equiv 0 \pmod{2}$, which implies $yz \in L$. Hence, $N_L(n)$ is bounded above by a certain absolute constant. \square

The notion of many-one 1-NLIN reducibility can be expanded into other complexity classes (such a complexity class is called *relativizable*). A language A is called *many-one low* for a relativizable complexity class \mathcal{C} of languages or of functions if $\mathcal{C}_m^A \subseteq \mathcal{C}$. A class \mathcal{D} is *many-one low* for \mathcal{C} if every set in \mathcal{D} is many-one low for \mathcal{C} . The notation $\text{low}_m \mathcal{C}$ denotes the collection of all languages that are many-one low for \mathcal{C} . For instance, Proposition 3 implies $\text{low}_m 1\text{-NLIN} = \text{REG}$. The following corollary is a direct consequence of Theorem 5.

Corollary 3. $\text{REG} = \text{low}_m 1\text{-}\#\text{LIN} = \text{low}_m 1\text{-GapLIN}$.

Proof. To prove the corollary, we first note that $\text{REG} \subseteq \text{low}_m 1\text{-}\#\text{LIN}$. Conversely, for any set A in $\text{low}_m 1\text{-GapLIN}$, since $\chi_A \in 1\text{-FLIN}_m^A \subseteq 1\text{-GapLIN}_m^A$, it follows that $\chi_A \in 1\text{-GapLIN}$. Thus, A is in 1-SPLIN , which equals REG . \square

We further introduce another counting class 1-C=LIN (pronounced “one C equal lin”) as the collection of all languages of the form $\{x \mid f(x) = 0\}$ for certain functions f in 1-GapLIN . The language $L_{eq} = \{a^n b^n \mid n \geq 0\}$, for instance, belongs to 1-C=LIN . The class 1-synC=LIN is the subset of 1-C=LIN defined only by linear-time synchronized counting 1TMs.

Proposition 5 leads to the following relationship between 1-C=LIN and 1-PLIN .

Lemma 9. $1\text{-C=LIN} \subseteq 1\text{-PLIN} \subseteq 1\text{-NLIN}_m^{1\text{-C=LIN}}$.

Similar to Lemma 6, we can prove that $\text{SL}_{rat}^{\overline{\overline{}}} \subseteq 1\text{-synC=LIN}$. Moreover, by Lemma 8, we can show that $1\text{-C=LIN} \subseteq \text{SL}_{rat}^{\overline{\overline{}}}$. Therefore, the following theorem holds.

Theorem 6. $1\text{-C=LIN} = 1\text{-synC=LIN} = \text{SL}_{rat}^{\overline{\overline{}}}$

Since $\text{SL}_{rat}^{\overline{\overline{}}}$, $\text{co-SL}_{rat}^{\overline{\overline{}}}$, and SL_{rat} are all different classes [18], we immediately obtain the following corollary.

Corollary 4. $1\text{-C=LIN} \neq \text{co-}1\text{-C=LIN} \neq 1\text{-PLIN}$.

Recall from Lemma 1 that the set L_{num} is not in REG/n . Obviously, this set belongs to 1-C=LIN . We can also prove that the non-context-free language $L_{abc} = \{a^n b^n c^n \mid n \in \mathbb{N}^+\}$ is in $1\text{-C=LIN} \cap \text{REG}/n$. Thus, we have the following separation.

Proposition 6. $1\text{-C=LIN} \not\subseteq \text{REG}/n$ and $1\text{-C=LIN} \cap \text{REG}/n \not\subseteq \text{CFL}$.

§7. Quantum Computation. The notion of a quantum Turing machine was introduced by Deutsch and reformulated by Bernstein and Vazirani [3]. In accordance with our definition of 1TMs, we use a slightly more general model of

one-tape quantum Turing machines with a tape head allowed to stay still. A *one-tape quantum Turing machine* (abbreviated 1QTM) is similar to the classical 1TM except that its transition function δ is a map from $Q \times \Gamma$ to the vector space $\mathbb{C}^{Q \times \Gamma \times \{L, N, R\}}$. For more detail, the reader refers to [3, 21, 22].

Let K be any nonempty subset of \mathbb{C} . A language L is in 1-BQLIN_K if there exist a linear-time¹¹ well-formed¹² stationary¹³ 1QTM M with K -amplitudes¹⁴ and an error bound $\varepsilon > 0$ such that, for every string x , (i) if $x \in L$, then M accepts input x with probability $\geq 1/2 + \varepsilon$ and (ii) if $x \notin L$, then M accepts input x with probability $\leq 1/2 - \varepsilon$.

Note that every reversible 1TM can be viewed as a well-formed stationary 1QTM with \mathbb{Q} -amplitudes. Proposition 2 therefore implies $\text{REG} \subseteq 1\text{-BQLIN}_{\mathbb{Q}}$. By applying the argument used in [1] for the containment $\text{BQP} \subseteq \text{PP}$, we can show that $1\text{-BQLIN}_{\mathbb{Q}} \subseteq 1\text{-PLIN}$ because the amplitude set is restricted to \mathbb{Q} .

Proposition 7. $\text{REG} \subseteq 1\text{-BQLIN}_{\mathbb{Q}} \subseteq 1\text{-PLIN}$.

A variant of quantum Turing machine, so-called a “nondeterministic” quantum Turing machine, which is considered as a quantum analogue of a nondeterministic Turing machine, was introduced by Adleman et al. [1]. Let K be any nonempty subset of \mathbb{C} . A language L is in 1-NQLIN_K if there exist a linear-time well-formed stationary 1QTM M with K -amplitudes such that, for every x , $x \in L$ iff M accepts input x with positive probability.

Brodsky and Pippenger [4] presented a measurement-once model of one-way quantum finite automaton that recognizes L_{eq} with unbounded-error probability. By a slight modification of their algorithm, we can show that $\overline{L_{eq}}$ belongs to $1\text{-NQLIN}_{\mathbb{Q}}$.

Theorem 7. $\text{REG} \subsetneq 1\text{-NQLIN}_{\mathbb{Q}}$.

Acknowledgments. The first author is grateful to the Japan Science and Technology Corporation and the 21st Century COE Security Program of Chuo University for the financial support. He also thanks Masahiro Hachimori for valuable suggestions on probabilistic computations. The second author thanks Harumichi Nishimura and Raymond H. Putra for discussions on Turing machines.

¹¹ The *running time* of M on input x is the minimal number t such that, at time t , all configurations of M on input x become configurations with halting states for the first time. Thus, any time-bounded 1QTM can be considered as synchronized 1TMs.

¹² A 1QTM M is *well-formed* if its time-evolution operator preserves the ℓ_2 -norm (Euclidean norm), where the *time-evolution operator* for M is the operator that maps a configuration to the next configuration resulted by the transition function δ of M .

¹³ A time-bounded 1QTM M is called *stationary* if, when M halts, the tape head halts in the same cell (not necessarily the start cell).

¹⁴ For any subset K of \mathbb{C} , a 1QTM is said to have K -amplitudes if all amplitudes in δ are drawn from K .

References

1. L. M. Adleman, J. DeMarrais, and M. A. Huang. Quantum computability. *SIAM J. Comput.*, **26** (1997), 1524–1540.
2. C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, **17** (1973), 525–532.
3. E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, **26** (1997), 1411–1473.
4. A. Brodsky and N. Pippenger. Characterizations of 1-way quantum finite automata. *SIAM J. Comput.*, **31** (2002), 1456–1478.
5. C. Damm and M. Holzer. Automata that take advice. *Proc. 20th MFCS*, Springer’s LNCS Vol.969, pp.149–158, 1995.
6. C. Dwork and L. J. Stockmeyer. A time complexity gap for two-way probabilistic finite state automata. *SIAM J. Comput.*, **19** (1990), 1011–1023.
7. C. Dwork and L. Stockmeyer. Finite state verifiers I: The power of interaction. *J. ACM*, **39** (1992), 800–828.
8. F. C. Hennie. One-tape, off-line Turing machine computations. *Inform. Control*, **8** (1965), 553–578.
9. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
10. R. M. Karp. Some bounds on the storage requirements of sequential machines and Turing machines. *J. ACM*, **14** (1967), 478–489.
11. J. Kaneps and R. Freivalds. Minimal nontrivial space complexity of probabilistic one-way Turing machines. *Proc. 19th MFCS*, Springer’s LNCS Vol.452, pp.355–361, 1990.
12. K. Kobayashi. On the structure of one-tape nondeterministic Turing machine time hierarchy. *Theor. Comput. Sci.*, **40** (1985), 175–193.
13. A. Kondacs and J. Watrous. On the power of quantum finite state automata. *Proc. 38th FOCS*, pp.66–75, 1997.
14. I. I. Macarie. Space-efficient deterministic simulation of probabilistic automata. *SIAM J. Comput.*, **27** (1998), 448–465.
15. P. Michel. An NP-complete language accepted in linear time by a one-tape Turing machine. *Theor. Comput. Sci.*, **85** (1991), 205–212.
16. M. O. Rabin. Probabilistic automata. *Inform. Control*, **6** (1963), 230–245.
17. P. Turakainen. On stochastic languages. *Inform. Control*, **12** (1968), 304–313.
18. P. Turakainen. On languages representable in rational probabilistic automata. *Annales Academiae Scientiarum Fennicae*, Ser.A **439** (1969), 4–10.
19. P. Turakainen. Generalized automata and stochastic languages. *Proc. Amer. Math. Soc.*, **21** (1969), 303–309.
20. T. Yamakami. Average case complexity theory. Ph.D. dissertation, University of Toronto, 1997. Technical Report 307/97, University of Toronto. See also ECCC Thesis Listings.
21. T. Yamakami. A foundation of programming a multi-tape quantum Turing machine. *Proc. 24th MFCS*, Springer’s LNCS Vol.1672, pp.430–441, 1999.
22. T. Yamakami. Analysis of quantum functions. To appear in *International Journal of Foundations of Computer Science*. A preliminary version appeared in *Proc. 19th FST&TCS*, Springer’s LNCS Vol.1738, pp.407–419, 1999.