Random Extraction from Freehand Drawings and Its Semantics Based on Forcing Complexity: Extended Abstract

Toshio Suzuki*and Akio Kawanishi
Department of Mathematics and Information Sciences,
Osaka Prefecture University, Sakai, Osaka 599-8531, Japan
toshios@acm.org whitecat@titan.ocn.ne.jp

November 24, 2005; revised Feburuary 20, 2006

Abstract. We present an algorithm that converts a set of bitmaps of freehand drawings to an almost random bit sequence. The goal of this article is to show, by introducing a mathematical model of an irregular object, that our algorithm preserves the irregularity of freehand drawings. Our model is defined by using forcing complexity, that is, the minimum size of a forcing condition. We show that our model is more suitable for our purpose than known models of weak random objects such as Martin-Löf randomness, resource bounded randomness and genericity, min-entropy and so on. We show that every Martin-Löf random oracle is irregular in our sense. We show experimental results, too.

1 Introduction

We present an algorithm that converts a set of bitmaps of freehand drawings to an almost random bit sequence (§3). Fig. 1 is an example of such a bitmap. We show statistical results (§4) for five sets of bitmaps. Each sets consists of particular type of bitmaps (Fig. 1–5). We expect our algorithm will produce good random seeds for other random number generators.

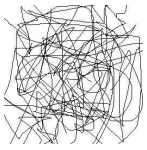

Figure 1:

Figure 2:

Figure 3:

Figure 4:

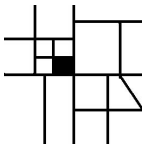

Figure 5:

*The author was partially supported by Grant-in-Aid for Scientific Research (No. 14740082 and No. 17540131), Japan Society for the Promotion of Science.

Although statistical tests are necessary to examine goodness of a random number generator, they are not sufficient. The reason is as follows. A generator of linear congruence method or of feedback shift register method [Ge03] often passes statistical tests with good scores. However, a "random" sequence given by such a method is not irregular at all in the sense of computational complexity. In particular, such a sequence has low Kolmogorov complexity, and has low circuit complexity. In order to show that an output sequence is enough irregular in the sense of computational complexity, we have to define mathematical model (in other words, semantics) of an irregular object, and have to show that an output is irregular under certain reasonable hypotheses. For a string u, its Kolmogorov complexity denotes, roughly speaking, the least length of a computer program that outputs the string. An infinite binary sequence is called Martin-Löf random if it passes all effective statistical tests. It is known as a result of Schnorr that an infinite binary sequence is Martin-Löf random if and only if every initial segment of it has large Kolmogorov complexity. For our purpose, the class of all Martin-Löf random oracles is a too small class as a model of the class of all irregular objects. Degree of irregularity of a freehand drawing is not necessarily so high as a Martin-Löf random oracle. On the other hand, the class of all oracles without polynomial-sized circuits is a too large class as a model of the class of all irregular objects. Because, even if an oracle X does not have polynomial-sized circuits, it is possible that $\forall n \in \mathbb{N}\, X(2n) = X(2n+1)$.

The goal of this article is to show that our algorithm preserves irregularity of freehand drawings, by giving semantics of irregular objects and random extraction. Our semantics is based on forcing complexity, that is, the minimal size of a forcing condition. Contrastively to Kolmogorov complexity, the smaller forcing complexity shows the higher degree of randomness. Forcing is a basic proof technique of set theory. In the theory of forcing, it is studied how properties of a large structure is determined by its partial information. Many applications of forcing to computational complexity are known. The forcing method that we use is a very simple one introduced by Dowd [Do92]. In this setting, the role of a large structure is played by an oracle, where an oracle denotes (the characteristic function of) a subset of $\{0,1\}^*$, and $\{0,1\}^*$ denotes the set of all strings of finite lengths. A function is called *a forcing condition* if its domain is a finite subset of $\{0,1\}^*$ and its range is a subset of $\{0,1\}$. By adding a set of query symbols $\{\xi^1, \xi^2, \xi^3, \cdots\}$ to the usual propositional calculus, we get *the relativized propositional calculus* [Do92]. For each n, the symbol ξ^n is an n-ary connective. If $q_1, q_2, \cdots, q_n$ are propositional variables, the interpretation of $\xi^n(q_1, q_2, \cdots, q_n)$ is, roughly speaking, given by "the string $q_1 \cdots q_n$ belongs to a given oracle". When a formula F of the relativized propositional calculus and an oracle X are given, the concept of "F is a tautology with respect to X" is naturally defined. If S is a forcing condition and F is a tautology with respect to every oracle X that is an extension of S, then we say that S *forces* F. Suppose that r is a positive integer. A formula of the relativized propositional calculus is called *an r-query formula* if it has just r-many occurrences of query symbols. An oracle A is called *an r-Dowd oracle* if there exists a polynomial p such that for every r-query formula F that is a tautology with respect to A, there exists

a forcing condition S such that S is a subfunction of A, the cardinality of the domain of S is at most $p(|F|)$, where $|F|$ denotes the binary length of F, and such that S forces F. It is known that for every positive integer r, the class of all r-Dowd oracles has Lebesgue measure one (sketch of a proof is in [Do92], and more rigorous proofs are in [Su01, Su02]). Though Dowd used terminology "an r-generic oracle" to denote an r-Dowd oracle, the concept of r-Dowd oracles is different from the concept of r-generic oracles in arithmetical forcing. Note that we use forcing and tautologies only for semantics. Our algorithm does not utilize these concepts.

In §5, we introduce three requirements that a model of the class of irregular objects should satisfy. And, we define our mathematical model of an irregular oracle as to be an oracle that is r-Dowd for all positive integer r. Many researchers have been formulated mathematical models of "an almost random object". In §6, we review some of such models: Martin-Löf randomness, resource bounded randomness of Schnorr and Lutz, resource bounded genericity of Ambos-Spies et al., pseudorandomness in Goldreich's book and min-entropy of Zuckerman, and so on. We show that these models do not satisfy the three requirements in §5. And, we show that every Martin-Löf random oracle is irregular in our sense. In §7, we prove that our model satisfies the three requirements in §5, and that our algorithm preserves irregularity of freehand drawings.

This article is based on our research report [KS05] and our talks in McMaster University [SK05a] and in Tokyo Metropolitan University [SK05b].

2 Notation and Basic Definitions

General Definitions. By $\mathbb{N}$ we denote the set of all non-negative integers. Strings are ordered by length-lexicographic order, and $(k+1)$st one is denoted by $z(k)$. For example, $z(0)$ is the empty string that is denoted by λ, $z(1)$ is 0, and $z(2), z(3), z(4)$ are $1, 00, 01$, respectively. A subset of $\{0,1\}^*$ is called an *oracle*. An oracle is identified with its characteristic function. The class of all oracles is called *Cantor space*. For a set A, its cardinality is denoted by $|A|$. A set $T \subseteq \{0,1\}^*$ is called *sparse* if there exists a polynomial p such that for every positive integer n, it holds that $|\{u \in \{0,1\}^{\leq n} : T(u) = 1\}| \leq p(n)$. For basic concepts of computational complexity including oracle Turing machines, see [BDG88, volume I]. Suppose that $\mathcal{C}$ is a computational complexity class such as P or NP. Then, an infinite subset X of $\{0,1\}^*$ is called *a $\mathcal{C}$-immune set* if X does not have an infinite subset that belongs to $\mathcal{C}$. If both X and the complement of X is $\mathcal{C}$-immune, then we say that X is *$\mathcal{C}$-bi-immune*. For the concept of immune sets, see [BDG88, volume II] or [YS05]. For two functions f and g, if the domain of f is a subset of the domain of g and if g agrees with f in the domain of f, then we say "f is *a subfunction* of g" or "g is *an extension* of f". Throughout the article, "a bitmap file" is often called simply "a bitmap".

Runs and rate of 1. A given string u of length n, by *the rate of 1 in u* we denote $|\{i \in \{0,1,\cdots,n-1\} : u(i) = 1\}|/n$. In a given string, if a consecutive portion consists of a same alphabet and the portion is not properly

included by other such a portion, then it is called a *run*. For example, a string $0^7\,1^5\,0^3\,1^6$ has four runs, 0^7, 1^5, 0^3 and 1^6, where for example 0^3 denotes 000. Suppose n_0 and n_1 are positive integers and we have n_0-many cards written '0' and n_1-many cards written '1', and we make a trial to arrange them in one line. In each result of a trial, let r denote the number of runs. It is known that probabilistic distribution of r is approximately normal distribution, provided that n_0, n_1 and the number of trials are sufficiently large, and provided that the way of arrangement is truly random. Let $n = n_0 + n_1$ and $\beta = n_1/n$. Then the mean value of r is approximately $2n\beta(1-\beta)$, and the variance (that is, σ^2) of r is approximately $4n\beta^2(1-\beta)^2$. Then, the probabilistic distribution of the following is approximately the standard normal distribution.

$$\frac{r - 2n\beta(1-\beta)}{2\sqrt{n}\beta(1-\beta)} \quad . \tag{2.1}$$

As is well-known, in the standard normal distribution, the mean value is 0, the variance is 1, and the interval $[-1.96, 1.96]$ has probability about 0.95.

The Relativized Propositional Calculus and Forcing Complexity. *The relativized propositional calculus* is defined as in Introduction section. Given an oracle A, we define an n-ary Boolean function A^n as follows, and ξ^n is interpreted as A^n. For each $k = 0, 1, \cdots, 2^n - 1$, the value of A^n at the $(k+1)$st n-bit string in lexicographic order is defined as to be $A(z(k))$. For example, $A^2(00)$, $A^2(01)$, $A^2(10)$ and $A^2(11)$ are $A(\lambda)$, $A(0)$, $A(1)$, $A(00)$, respectively. The concepts of *a forcing condition*, *"to force"* and *"a tautology with respect to a given oracle"* are defined as in Introduction section. For example, let F be the following formula of the relativized propositional calculus. $(q_0 \Leftrightarrow \xi^3(q_1, q_2, q_3)) \Rightarrow (q_4 \vee \neg q_0)$. F is a tautology with respect to (the characteristic function of) the empty set. Suppose that S is a constant function with domain $\{z(0), z(1), \cdots, z(7)\}$ and with value 0. Then S is a forcing condition, and S forces F. No proper subfunction of S forces F. For each positive integer r, the concept of r-Dowd oracles is defined as in Introduction section. For more detailed explanation on the relativized propositional calculus, forcing complexity and Dowd-type generic oracles, see [Su01, Su02].

3 Our Algorithm

Description of Our Algorithm. In practical applications, we make sufficiently many monochrome bitmap files X_0, X_1, X_2, $\cdots$ by drawing, and we then convert them into Boolean matrices $Y^{(0)}$, $Y^{(1)}$, $Y^{(2)}$, $\cdots$. We assume that for each j, the bitmap X_j has a square canvas of N^2 pixels, and that $Y^{(j)}$ is of type $N \times N$, where N is a fixed positive integer. And, each black (white) pixel in a monochrome bitmap is converted to 1 (0) in the corresponding matrix. Throughout this section, C_1 and C_2 are fixed positive integers. We define the values of C_1 and C_2 depending on N. C_3 is defined as to be $\mathrm{floor}(\log_2(C_1 + C_2))$, where $\mathrm{floor}(x)$ denotes the maximum number not greater than x. Let q_1 be the quotient of N^2 divided by C_1. Our algorithm is an oracle Turing machine, and

the oracle is given by the sequence of matrices $Y^{(0)}$, $Y^{(1)}$, $Y^{(2)}$, $\cdots$. Let n be the quotient of q_1C_3 divided by 2, and ℓ be its remainder. An input of our algorithm is a non-negative integer, say $k = qn + k'$ (where $q, k' \in \mathbb{N}$ and $k' < n$), and output for k is given by the output of Algorithm 1 for the input k' and an oracle matrix $Y^{(q)}$.

Algorithm 1 /* The core of our algorithm. Output is a bit (0 or 1). */
input, an integer k $(0 \leq k < q_1C_3)$;
input, a Boolean matrix $A = (a_{i,j})$ $(i, j \in \{0, 1, \cdots, N-1\})$;
/* A plays a role of a finite portion of an oracle. */
STEP1: Define an array a as follows.
$a = a_{0,0} \cdots a_{0,N-1} a_{1,0} \cdots a_{1,N-1} \cdots a_{N-1,0} \cdots a_{N-1,N-1}$.
STEP2: Partition the array a so that each portion consists of C_1 bits. Let $a^{(0)} = a_0^{(0)} \cdots a_{C_1-1}^{(0)}$, $a^{(1)} = a_0^{(1)} \cdots a_{C_1-1}^{(1)}$, $\cdots$ be these portions. Let $r_0^{(0)}$, $r_1^{(0)}$, $\cdots$, $r_j^{(0)}$, $\cdots$ be the lengths of runs in $a^{(0)}$. For each j, let $r_j^{(0)\prime}$ be the binary expression of $r_j^{(0)} + C_2$. For each j, remove the leftmost 1 from $r_j^{(0)\prime}$, and let $s_j^{(0)}$ be the resulting string. Get an array by concatenating $s_0^{(0)}, s_1^{(0)}, \cdots, s_j^{(0)}, \cdots$, and let an array $b^{(0)}$ be its leftmost C_3 bits.
/* Example. Suppose that $C_2 = 2$, $C_3 = 5$ and $a^{(0)} = 0001111001\cdots$. Then $r_0^{(0)} = 3$, $r_1^{(0)} = 4$, $r_3^{(0)} = 2$, $r_0^{(0)\prime} = 101$, $r_1^{(0)\prime} = 110$ and $r_3^{(0)\prime} = 100$. And, $s_0^{(0)} = 10$, $s_1^{(0)} = 01$ and $s_3^{(0)} = 00$. Finally, we get $b^{(0)} = 10010$. */
By applying the same operation to $a^{(1)}, a^{(2)}, \cdots$, we get arrays $b^{(1)}, b^{(2)}, \cdots$. We concatenate $b^{(0)}, b^{(1)}, b^{(2)}, \cdots$ and let $b = b_0 \cdots b_{2n-1+\ell}$ be the resulting array (Fig. 6). /* For the definitions of n and ℓ, see the begining of this section. */
STEP3: Output $b_k \,\mathrm{XOR}\, b_{2n-k-1+\ell}$, where XOR denotes exclusive-or.
End, Algorithm 1.

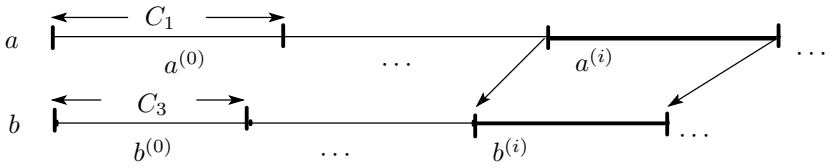

Figure 6: Block wise operation

Efficiency of Labor Using Our Algorithm. In order to make sufficiently large table of random numbers by using Algorithm 1, we have to draw many bitmap files. For simplicity, we assume that the volume of labor to draw is measured by the amount of pixels of canvases in bitmaps. Then volume of labor to draw M-many bitmaps, where each bitmap has a square canvas of N^2 pixels, is N^2M. When we use (matrices made of) such M-many bitmaps as oracles, the amount of outputs (the number of bits) is $(1/2)C_3 \,\mathrm{floor}(N^2/C_1)M$.

In the case of $N = 256$, $C_1 = 60$ and $C_2 = 2$, the value of the above formula is approximately $0.0417 \times 256^2 M$. This is the amount of harvests, and is a linear function of the volume of labor. We defined C_3 as to be $\log_2(C_1 + C_2)$. One may criticize our algorithm for the smallness of C_3. However, such criticism is valid only in the case where $N^2 M$ is not so larger than C_1 and C_2. Note that the amount of harvests is not logarithm of the volume of labor.

4 Results of Experiments

We implemented Algorithm 1 by the C++ programming language with $N = 256$, $C_1 = 60$ and $C_2 = 2$ (thus $C_3 = 5$). Table 1 summarizes our experiments.

file set	F_1	F_2	F_3	F_4	F_5
pen	mouse	pen tablet	mouse	pen tablet	mouse
motif	meaningless curves	meaningless curves	cartoons of faces	cartoons of faces	geometrical designs
m(β)	0.500	0.497	0.480	0.497	0.469
$\sigma^2(\beta)$	8.47 E -5	9.15 E -5	2.29 E -4	1.36 E -4	1.14 E -3
m(V)	-0.140	9.54 E -3	-0.186	-0.0853	-0.844
$\sigma^2(V)$	0.952	1.015	1.528	0.998	5.466
χ^2	pass	pass	pass	pass	fail
α	0.930	0.998	0.100	0.371	0.0000

Table 1: The symbol "E k", where k is an integer, denotes "$\times 10^k$ ".

For each $i = 1, \cdots, 5$, the bitmap file set F_i consists of 100 files, where 50 of them are drawn by the first author and other 50 are drawn by the second author. Fig. 1–5 in Introduction section are examples of members of F_1–F_5, respectively. Each bitmap in these five sets is drawn in about one minute. In the line named "pen", the item "mouse" denotes that the bitmaps are drawn by using mouse and a paint software (Paint of Microsoft Windows®). And, the item "pen tablet" denotes that the bitmaps are drawn by using a pen tablet (WACOM intuos 3® PTZ-630) that responds to stress of a pen. In the rest of this section, let $n = 2730$. By an execution of the C++ program, each bitmap is converted to an array of length n, that is, the array $\langle b_k \,\mathrm{XOR}\, b_{2n-k-1} : k = 0, 1, \cdots, n-1 \rangle$. Thus, each set of bitmaps is converted to a set of arrays. For each Boolean array u of length n, by β we denote the rate of 1 (see §2). And, by r we denote the number of runs in u. Moreover, we define V as follows (see equation (2.1) in §2). $V = (r - 2n\beta(1-\beta))/(2\sqrt{n}\beta(1-\beta))$. In the table, m($\beta$), $\sigma^2(\beta)$, m(V), $\sigma^2(V)$ denote mean value of β, variance of β, mean value of V and variance of V in the corresponding set of arrays, respectively. The line named χ^2 denotes results of chi-square goodness of fit test of V to the standard normal distribution. The degree of freedom in our test is 19. Throughout the rest of this section, we denote it by d: thus $d = 19$. We partition the real line into $(d+1)$-many intervals I_j $(j = 1, \cdots, d+1)$, symmetrically with respect to the origin O,

so that each I_j has probability $1/(d+1)$ in the standard normal distribution. Then our null hypothesis is: "For each j, among given 100 samples of statistic V, just $100/(d+1)$-many of them belongs to I_j." We test this hypothesis with significance level 0.05 (that is, 5 %). Let c be the value of chi-square computed by using the observed frequencies. Let α be the probability of the event "$x \geq c$" in the chi-square distribution with degree of freedom d. If $\alpha \geq 0.05$, the null hypothesis is not rejected. In this case, in the line named χ^2 of the table, we write "pass". If $\alpha < 0.05$, the null hypothesis is rejected. In this case, in the table, we write "fail". About introduction to chi-square tests, see for example [HMC05].

5 Mathematical Model of Irregular Objects

We do not introduce a mathematical model of an irregular string of finite length. Instead, we introduce a model of an irregular string of infinite length. In practical applications, we make sufficiently many bitmap files X_1, X_2, X_3, $\cdots$ by drawing, and converted them into matrices $Y^{(1)}$, $Y^{(2)}$, $Y^{(3)}$, $\cdots$, and again converted them into strings $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, $\cdots$ (see STEP 1 of Algorithm 1). By concatenating these strings, we get a string of enough long length. In theoretical investigation, we approximate such a long string by a string of infinite length. From experiences, we make the following working hypotheses.

Hypothesis 1: In each string $y^{(j)}$, the rate of 1 (see §2) may be far from 0.5 unless a drawer intentionally make the rate close to 0.5.

Hypothesis 2: It is impossible to make a fast computer program that correctly predicts strings $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, $\cdots$.

Hypothesis 3: However, the sequence gained by concatenating strings $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, $\cdots$ may have an infinite subsequence whose bits are predictable with high probability. For example, the four corners of a square canvas of a bitmap is often remain 0 (white) unless a drawer intentionally make the corners 1 (black).

We simplify the above three hypotheses, and formulate the following three requirements for our mathematical model of "irregular oracles". Note that they are not requirements for each individual irregular oracle but requirements for the class of all irregular oracles. Requirements 1 and 2 are mandatory, and Requirement 3 is optional.

Requirement 1 : For every positive integer m, there exists an irregular oracle X such that

$$\lim_{n\to\infty} \frac{|\{i < n : X(z(i)) = 1\}|}{n} \tag{5.1}$$

is at most $1/2^m$. And, there exists an irregular oracle X such that the limit (5.1) is no less than $1 - 1/2^m$ (see §2 for the symbol $z(i)$).

Requirement 2 : 1. An irregular oracle X does not have polynomial-sized circuits ($X \notin \mathrm{P/poly}$). Hence, it is not polynomial-time computable ($X \notin \mathrm{P}$).

2. Suppose that k is a positive integer, f is a function from $\{0,1\}^k$ onto $\{0,1\}$, and X is an irregular oracle. Then, there exists a natural number i such

that the following holds: $f(X(z(i)), X(z(i+1)), \cdots, X(z(i+k))) = 0$.
Requirement 3 : Suppose that A is an irregular oracle, $f : \{0,1\}^* \to \{0,1\}$ is a polynomial-time computable function, and $T \subseteq \{0,1\}^*$ is a sparse set such that T is polynomial-time computable. Let B be an oracle defined as follows. For each string u, if $u \in T$ then $B(u)$ is defined as to be $f(u)$, and otherwise $B(u)$ is defined as to be $A(u)$. Then, B is irregular. (For sparse sets, see §2.)

Now, we introduce our mathematical model of irregular oracles. If an oracle D has the following Property 1, we consider that D is irregular.
Property 1 "D is an r-Dowd oracle for any positive integer r."

Suppose that f is a function from Cantor space to Cantor space such that there exists a polynomial-time-clocked oracle Turing machine $M^{\sim}$ such that for any oracle A and for any string u, it holds that $M^A(u) = f(A)(u)$. If f has the following Property 2, then we consider that f is a function that preserves irregularity. The class of all such functions is closed under composition.

Property 2 "For any oracle D that has Property 1, $f(D)$ has Property 1, too."

6 Comparison of Our Model with Other Models

Many researchers have been formulated mathematical models of "an almost random string". In this section, we review some of such models and show that they do not satisfy the three requirements in §5.

Martin-Löf Randomness. In the following, open set is that of canonical product topology of Cantor space.

Definition 1 [YDD04] (see [Ca02, ML66] for more formal definition)

1. A computable collection $\{V_n : n \in \mathbb{N}\}$ of computably enumerable open sets is said to be a *Martin-Löf test* if for all n the measure of V_n is at most 2^{-n}.
2. An oracle X is said to *pass* the Martin-Löf test if $\exists\, n\, X \notin V_n$.
3. An oracle X is said to be *Martin-Löf random* if it passes all Martin-Löf test.

The class of all Martin-Löf random oracles has Lebesgue measure one. For a string u, let $K(u)$ denote the (prefix free) Kolmogorov complexity of u. Schnorr (1971) showed that an oracle X is Martin-Löf random if and only if

$$\exists c \in \mathbb{N}\, \forall n \in \mathbb{N}\, K(X \upharpoonright \{0, 1, \cdots, n-1\}) \geq n - c \ .$$

If X is Martin-Löf random then the limit (5.1) in §5 is 1/2, and X is P-bi-immune. Hence, if we replace "an irregular oracle" in the statements Requirement 1 and 3 by "a Martin-Löf random oracle", resulting statements do not hold.

Theorem 1 *Suppose that an oracle A is Martin-Löf random. Then A is r-Dowd for any positive integer r.*

Resource Bounded Randomness and Genericity. Schnorr (1971) criticized the concept of Martin-Löf randomness, and proposed concepts of resource bounded measure and resource bounded randomness. Lutz (1992) rediscovered and developed these concepts. Resource bounded randomness has close relationships with resource bounded genericity of Ambos-Spies et al. For a survey of resource bounded randomness and genericity, see [AM97].

Suppose X is an oracle and $t(n)$ is a numerical function such that $\forall n \in \mathbb{N}\, t(n) \geq n^2$. Then, every $t(n)$-random oracle is $t(n)$-generic [ANT96, ATZ97], and every $t(n)$-generic oracle is DTIME($t(2^{n-1})$)-bi-immune [ANT96]. Hence, if we replace "an irregular oracle" in the statement of Requirement 3 by "an oracle that is p-generic for every polynomial p", then resulting statement fails. The argument for p-randomness is parallel.

Time Complexity, Circuit Complexity and Their Variants. Even if we assume that an oracle X has enough high time complexity, it is possible that $\forall n \in \mathbb{N}\, X(2n) = X(2n+1)$. Therefore, if we replace "an irregular oracle" in the statement of Requirement 2 by "an oracle that is not polynomial-time computable", then resulting statement fails. The argument for circuit complexity is parallel. It is known that no 1-Dowd oracle has polynomial-sized circuits [Su02].

Pseudorandomness in Goldreich's Book. In the study of cryptography by means of computational complexity, the concept of pseudorandomness is defined as a probabilistic distribution that is not distinguishable from uniform distribution by a computer program with certain restriction on resources (time, memory and so on) [Go99, Chapter 3]. If we replace "an irregular oracle" in the statement of Requirement 1 by "a pseudorandom oracle" in the above sense, then resulting statement does not hold.

Min-entropy. Zuckerman introduced the concept of min-entropy (1990) in order to formulate a mathematical model for a physical random source. The concept of extractor introduced by Nisan and Zuckerman (1993, 1996) is based on min-entropy. For a survey on their theory, see [NTs99, Sh02]. Min-entropy is not suitable for a model of an irregular objects in our setting. The reason is similar to the case of pseudorandomness in Goldreich's book.

7 Semantics of Random Extraction

Theorem 2 *The statements of Requirement 1, 2, 3 of §5 holds if we replace "an irregular oracle" by "an oracle that is r-Dowd for all positive integer r".*

We define an oracle Turing machine M as follows. By using the machine M, we introduce a mathematical model of Algorithm 1, and we show that the algorithm preserves irregularity of freehand drawings. Suppose that N, C_1, C_2 are positive integers, and let $C_3 = \mathrm{floor}(\log_2(C_1 + C_2))$, where we assume $C_2 < C_1 < N$. And, let q_1 be the quotient of N^2 divided by C_1. Let n be the

quotient of q_1C_3 divided by 2, and ℓ be its remainder. For an oracle X and an input string $z(k)$, where $k \in \mathbb{N}$, we define the output of M as follows. Let q be the quotient of k divided by n, and let k' be its remainder. Let a be the string $X(q_1C_1q)X(q_1C_1q+1)\cdots X(q_1C_1(q+1)-1)$. We execute STEP 2 of Algorithm 1 for a. Next, we execute STEP 3 of Algorithm 1 with k' (for k in STEP 3), and get an output.

Let f denote the mapping of an oracle such that for each oracle X, $f(X)$ is (the characteristic function of) the oracle $\{u \in \{0,1\}^* : M^X(u) = 1\}$. The oracle X is a mathematical model of (an array given by) sufficiently many bitmaps, provided that X has Property 1 of §5. And, the function f is a mathematical model of Algorithm 1.

Theorem 3 *The mapping f defined above has Property 2 of §5. That is, if an oracle A is r-Dowd for all positive integer r then so is $f(A)$.*

Acknowledgment and final note. We thank Professor Teruhisa Hochin at Osaka Prefecture University for his advice on how to deal with bitmap files. Forthcoming complete version of this article will contain all the proofs that are omitted from this extended abstract and will contain more detailed experimental results.

References

[AM97] Ambos-Spies, K., Mayordomo, E.: Resource-bounded measure and randomness. In: Lecture Notes in Pure and Applied Mathematics **187** (A. Sorbi, Eds.), pp.1-47, Marcel Dekker, New York, 1997.

[ANT96] Ambos-Spies, K., Neis, H.-C. and Terwijn, S. A.: Genericity and measure for exponential time. *Theoret. Comput. Sci.*, **168** (1996), pp. 3-19.

[ATZ97] Ambos-Spies, K., Terwijn, S. A. and Zheng, X.: Resource bounded randomness and weakly complete problem. *Theoret. Comput. Sci.*, **172** (1997), pp. 195-207.

[BDG88] Balcázar, J. L., J. Díaz, and J. Gabarró: *Structural complexity I & II.* Springer, Berlin, 1988 (I), 1990 (II).

[Ca02] Calude, C.S.: *Information and Randomness: An Algorithmic Perspective, 2nd ed.*. Springer, Berlin / Tokyo, 2002.

[Do92] Dowd, M.: Generic oracles, uniform machines, and codes. *Information and Computation*, **96** (1992), pp. 65-76.

[Ge03] Gentle, J. E.: *Random number generation and Monte Carlo methods* (*second edition*). Springer, New York, 2003.

[Go99] Goldreich, O.: *Modern cryptography, probabilistic proofs and pseudorandomness*. Springer, Berlin / New York, 1999.

[HMC05] Hogg, R. V., McKean, J. W., Craig, A. T.: *Introduction to Mathematical Statistics, sixth edition*. Pearson Education, London, 2005.

[KS05] Kawanishi, A. and Suzuki, T.: Random extraction from freehand drawings and its semantics: preliminary report (in Japanese). In: Sūrikaisekikenkyūsho Kōkyūroku **1442** (2005), pp. 8-41.

[ML66] Martin-Löf, P.: The definition of random sequences. *Information and Control*, **9** (1966), pp. 602-619.

[NTs99] Nisan, N. and Ta-Shma, A.: Extracting randomness: a survey and new constructions. *Journal of Computer and System Sciences*, **58** (1999), pp. 148-173.

[Sh02] Shaltiel, R.: Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, **77** (2002), pp. 67-95.

[Su01] Suzuki, T.: Forcing complexity: minimum sizes of forcing conditions. *Notre Dame J. Formal Logic*, **42** (2001), pp. 117-120.

[Su02] Suzuki, T.: Degrees of Dowd-type generic oracles. *Inform. and Comput.*, **176** (2002), pp. 66-87.

[SK05a] Suzuki, T. and Kawanishi, A.: Random extraction from freehand drawings and its semantics based on forcing complexity. A talk at *Franco-Canadian Workshop on Combinatorial Algorithm (COMAL 2005)*, McMaster University, Hamilton, Canada, August 18-20, 2005.

[SK05b] Suzuki, T. and Kawanishi, A.: Random extraction from freehand drawings and its semantics based on forcing complexity. A talk at *6th symposium on algebra and computation (AC2005)*, Tokyo Metropolitan University, Japan, November 15-18, 2005.

[YDD04] Yu, L. and Ding, D. and Downey, R.: The Kolmogorov complexity of random reals. *Annals of Pure and Applied Logic*, **129** (2004), pp. 163-180.

[YS05] Yamakami, T. and Suzuki, T.: Resource bounded immunity and simplicity, *Theoret. Comput. Sci.*, **347** (2005), pp. 90-129.