Indistinguishability

Sophie Laplante*
University of Chicago

John Rogers†
DePaul University

December 8, 1996

Abstract

Two strings are *distinguished* by a program if the program always outputs one of $\{0, 1, \perp\}$, and it outputs 1 on one of the strings and 0 on the other. Two strings are k-indistinguishable if no program of length at most k can distinguish them.

In this paper, we introduce the formal definitions for indistinguishability, and prove some basic properties. Indistinguishability is compelling notion, and we illustrate this by showing that our framework provides a "dual" perspective on nondeterministic communication complexity. We do this by giving a tight upper and lower bound on nondeterministic communication complexity in terms of distinguishability.

Introduction

How hard is it to decide an instance of a language A? This question has always been at the center of complexity theory. Complexity cores, defined by Lynch [Ly75] and studied by several other authors [OS86, BD87, etc.] provided complexity theorists with a new angle: Are there particular instances which make a language hard to decide, independently of the algorithm chosen to solve it? Instance complexity was defined [OKSW94] and studied [BO94, FK95, K95] with a somewhat similar intention: quantifying the hardness of individual instances with respect to a set, independently of the algorithm chosen to solve it.

In this paper, we take the approach of instance complexity a step further, and present what we hope will provide a new perspective on the hardness of deciding languages. We tackle the problem of hardness in a way that is *language independent*. As in instance complexity, we quantify how hard it is to distinguish two strings, and prove that there in fact are infinitely many *pairs* of strings, even *sets* of strings, that are *indistinguishable*. Therefore whenever one of these strings is in an "easy" language, then so must be all of the strings that are indistinguishable from it.

The framework of distinguishability lends itself particularly well to the notion of nondeterministic communication complexity. We show that the two notions are connected in a strong way by providing an upper and lower bound on nondeterministic communication complexity in terms of distinguishability.

*University of Chicago, Department of Computer Science, 1100 E. 58th St., Chicago, IL 60637 USA. Email: laplante@cs.uchicago.edu.

Supported in part by NSF grant CCR 92-53582.

†DePaul University, School of Computer Science, 243 S. Wabash Ave.,Chicago IL 60604 USA. Email: rogers@cs.depaul.edu.

The definition of indistinguishability has the same flavor as Kolmogorov complexity and instance complexity in that the focus is on the *length of the shortest program* which attempts to distinguish strings from one another. Section 1 lays out the definitions and notation used throughout the paper. In section 2 we introduce distinguishability and give an invariance theorem for distinguishability. Section 3 contains some basic results that provide motivation for studying indistinguishability. Section 4 links distinguishability with instance complexity and CD complexity. Finally, section 5 establishes a relationship between communication complexity and distinguishability.

1 Preliminaries

This section will provide the necessary background for the paper, including Kolmogorov complexity, CD complexity, and instance complexity.

1.1 Notation

Our model of computation is the standard two-tape Turing machines which take input in $\{0,1\}^*$ and output either 0, 1, or $\perp$. Strings are usually represented by variables w, x, y, sets of strings by $A, B, \ldots$, string lengths by $|x| = n$, and time bounds by t. Uppercase Greek letters, such as Φ, represent partial recursive functions. $P^{\leq k}$ denotes the set of programs of length at most k, and N_k denotes $|P^{\leq k}| = 2^{k+1} - 1$, so that $P^{\leq k} = \{p_1, \ldots p_{N_k}\}$.

1.2 Kolmogorov complexity

We use a few elementary definitions and results from Kolmogorov complexity.

Definition 1.1 *Fix a universal Turing machine Φ. For any pair of strings $x, y \in \{0,1\}^*$, the Kolmogorov complexity of x relative to y is defined as*

$$C_\Phi(x|y) = Min\{|p| : \Phi(p, y) = x\}.$$

For the remainder of this discussion, we fix Φ and write $C(x|y)$ instead of $C_\Phi(x|y)$. When $y = \epsilon$, the empty string, we simply write $C(x)$.

Proposition 1.1 *For every n, there is a string x of length n with $C(x) \geq n$.*

A string x with this property is called *incompressible*.

The reader is referred to the textbook by Li and Vitányi [LV93] for further background on Kolmogorov complexity.

1.3 CD complexity

In the context of time-bounded computation, it makes sense to make the distinction between programs that print out a string x and those that accept exactly the string x. The latter is the object of CD complexity, or distinguishing complexity.

Definition 1.2 *Fix a universal Turing machine Φ which runs in time bounded by $t(|p|,|x|)$ and whose output is 0 or 1. For any pair of strings $x, y \in \{0,1\}^*$, the t-time-bounded Kolmogorov distinguishing complexity of x relative to y is defined as*

$$CD^t_\Phi(x|y) = Min\{|p| : \Phi(p,y,z) = 1 \textit{ iff } z = x\}.$$

1.4 Instance complexity

Instance complexity [OKSW94] studies the complexity of solving specific instances of a language A. The measure used is the length of the shortest "special-purpose" program which determines correctly whether the given instance is, or is not, in A. By "special-purpose", we mean that the only programs considered are the ones which always output one of 0,1, or $\perp$, and are *consistent* with A, meaning when the output is not $\perp$ ("I don't know!"), the answer must be correct. A formal definition follows.

Definition 1.3 *[OKSW94] Fix a universal Turing machine M with output symbols $\{0,1,\perp\}$, which computes the partial recursive function Φ. Fix also a recursive set A.*

1. *A Φ-program p is* consistent *with A if for all x, when $\Phi(p,x) \neq \perp$ then in fact $\Phi(p,x) = \chi_A(x)$.*
2. *The* instance complexity *of x relative to the set A, written $ic(x : A)$, is the length of the shortest program p consistent with A and for which $\Phi(p,x) \neq \perp$.*

2 Indistinguishability: strings attached

In this section we give the definition of distinguishability, and we prove some basic properties about distinguishable strings.

2.1 Distinguishability

In order for our notion of indistinguishability to be robust, the programs we consider first are ones realized by a fixed partial recursive function Φ. When we write $p(x)$ for a program p and input x, we mean the output of Φ on input $\langle p, x\rangle$, where $\langle\cdot,\cdot\rangle$ is a standard pairing function.

The output alphabet of Φ is $\{0,1,\perp\}$, and the programs we consider always output one of $\{0,1,\perp\}$. We mean $\perp$ to represent "I don't know"; 0 and 1 might represent the characteristic function of some unspecified set. We say that an algorithm *distinguishes* two strings if it outputs 0 on one string and 1 on the other. If Φ runs in time bounded by the function $t(|p|,|x|)$, then we say the strings are (k,t)-distinguishable. A formal definition follows.

Definition 2.1 *Two strings $x \neq y$ are (Φ, t, k)-distinguishable if there is a t-time-bounded program p for Φ of length k such that $p(x) \neq \perp, p(y) \neq \perp, p(x) \neq p(y)$.*

In the time-unbounded case, we write x and y are (Φ, k)-distinguishable.

Definition 2.2 *$Dist^t_\Phi(x, y)$ is the smallest k such that x, y are (Φ, t, k)-distinguishable. If $x = y$, then $Dist^t_\Phi(x, y) = \infty$.*

2.2 Invariance

We now show that the notion of indistinguishability is independent of the programming system Φ.

Theorem 2.1 *There is a universal partial recursive function Φ_0 such that for any partial recursive Φ, time bound t, and pair of strings x, y,*

$$Dist^{\tilde{t}}_{\Phi_0}(x, y) \leq Dist^t_\Phi(x, y) + c'$$

where $\tilde{t} \leq ct(n)\log(t(n))$, and c, c' are constants depending only on Φ.

Proof: This is a standard argument, based on efficient simulation of multi-tape Turing machines by a two-tape Turing machine [LV93, page 378]. Let $M_1, M_2, \ldots$ be a standard enumeration of multitape Turing machines, and let $\Phi_1, \Phi_2, \ldots$ be the corresponding partial recursive functions. Using the standard simulator U, we know that for each partial recursive function Φ_i, $U(\langle\langle i, \cdot\rangle, \cdot\rangle)$ computes the same function as $\Phi_i(\langle \cdot, \cdot\rangle)$, and if $t(n)$ is a bound on the running time of $\Phi_i(\langle p, \cdot\rangle)$, the running time of $U(\langle\langle i, p\rangle, \cdot\rangle)$ is bounded by $\tilde{t}(n) = ct(n)\log(t(n))$, where c is a constant depending only on i. Furthermore, if we let Φ_0 be the partial function computed by the Turing machine U,

$$\mathrm{Dist}^{\tilde{t}}_{\Phi_0}(x, y) \leq \mathrm{Dist}^t_{\Phi_i}(x, y) + c',$$

where c' also depends only on i. This is because a Φ_i-program p which distinguishes x, y can be rewritten into a Φ_0-program to distinguish the same x, y, namely $\langle i, p\rangle$ $\Box$

Given the invariance of Dist under the choice of Φ, we can write x and y are (t, k)-distinguishable, or simply k-distinguishable in the time-unbounded case. Also we write $\mathrm{Dist}^t(x, y)$ to mean $\mathrm{Dist}^t_\Phi(x, y)$ for a fixed universal partial recursive function Φ.

2.3 Indistinguishable pairs of strings

It is easy to see that for any $x \neq y, |x| = |y| = n, \mathrm{Dist}^t_\Phi(x, y) \leq \log(n) + c$, since a program need only specify a bit position in which the two strings differ.

The following lemma establishes that for every string length, there are pairs of distinct strings x, y of length n such that $\mathrm{Dist}^t_\Phi(x, y) \geq \log(n) - 1$.

Lemma 2.1 *$\forall n, k, 0 < k < \log(n)$, there are at least $2^{2n-2^{k+1}} - 2^{n-1}$ pairs $x \neq y, |x| = |y| = n$ such that $Dist^t_\Phi(x, y) \geq k$*

Proof: To each input x of length n corresponds a list of the output of all the different programs $p \in P^{\leq k}$, i.e, $(p_1(x), \ldots, p_{N_k}(x))$. We call this list the *signature* of x (under $P^{\leq k}$).

We say x's signature is *compatible* with a binary string, or *tag*, $w = w_1 \cdots w_{N_k} \in \{0,1\}^{N_k}$ if whenever $p_i(x) \neq \perp, p_i(x) = w_i$. Notice that any 2 inputs x, y whose signatures are both compatible with the same tag w have $\mathrm{Dist}^t_\Phi(x, y) \geq k$. When this happens, we say that x and y *collide* on w.

From this point we use a simple counting argument to complete the proof. Since there are only N_k programs, there are only 2^{N_k} tags in $\{0,1\}^{N_k}$ and 2^n inputs competing for them. Even if we assume each signature is compatible with only one tag, the best case scenario (minimizing collisions) occurs when there is an equal number of collisions for each signature. Hence the number of collisions is bounded below by the following expression.

$$\begin{aligned} 2^{N_k}\binom{2^{n-N_k}}{2} &= 2^{N_k-1}2^{n-N_k}(2^{n-N_k}-1) \\ &= 2^{n-1}(2^{n-N_k}-1) \\ &= 2^{2n-N_k-1} - 2^{n-1} \\ &= 2^{2n-2^{k+1}} - 2^{n-1} \end{aligned}$$

□

When $k = \log(n) - 1$, the number of pairs is at least 2^{n-1}; when $k = \log(n) - 2$, the number is $2^{\frac{3n}{2}} - 2^{n-1}$, and when $k = c$ for a constant c, i.e., when we consider only constant-size programs, the number is $\frac{1}{2^{c+1}}2^{2n} - 2^{n-1}$, a considerable fraction of the total number of pairs of distinct strings of length n: $2^{2n} - 2^n$.

3 Indistinguishable tangles

From the proof of lemma 2.1, we get not only that there are many pairs of indistinguishable strings, but that there must be relatively large sets of strings compatible with a single tag w. By the pigeonhole principle, it is easy to see that there are always such "tangles" of size $2^{n-N_k} = 2^{n-2^{k+1}+1}$.

Definition 3.1 *A (Φ, t, k)-tangle T is a set of strings with the property that $\forall x, y \in T, Dist^t_\Phi(x, y) > k$. We say T is a (t, k)-tangle when Φ is a fixed universal partial recursive function.*

We can say more about the size and distribution of the tangles if we relate them to the Kolmogorov complexity of the strings involved. We prove in the next lemma that any string of high Kolmogorov complexity must be involved in a large tangle.

Lemma 3.1 *$\forall n, k, 0 < k < \log(n), |x| = n$, there is a (t,k)-tangle T_x including x with*

$$|T_x| \geq 2^{C(x)-2^{k+1}-1-\log(k)-O(\log(n))} = \frac{2^{C(x)-2^k}}{n^{O(1)}}$$

Proof: We bound the Kolmogorov complexity of a string x by encoding the following information:

- the values k and n
- a tag w compatible with x's signature under $P^{\leq k}$
- its index among inputs whose signatures are compatible with w.

We must also encode the lengths of the segments so they can be decoded correctly. For this we need an additional term of $O(\log(n))$ to encode the length of k and n. (Note that the length of w depends only on k so it is not necessary to encode any further information.)

As in the proof of Lemma 2.1, $P^{\leq k}$ is the set of programs of length at most k. Let N_k represent $|P^{\leq k}| = 2^{k+1} - 1$.

We get the following bound on $C(x)$.

$$\begin{aligned} C(x) &\leq N_k + \log(|T_x|) + \log(k) + O(\log(n)) \\ \log(|T_x|) &\geq C(x) - N_k - \log(k) - O(\log(n)) \\ &= C(x) - 2^{k+1} - 1 - \log(k) - O(\log(n)) \end{aligned}$$

So $|T_x| \geq 2^{C(x)-2^{k+1}-1-\log(k)-O(\log(n))}$, as claimed. □

To put this in perspective, it in interesting to look at the size of tangles associated with Kolmogorov-incompressible strings. If we allow programs of length up to $\log(n) - c$ for a constant $c > 1$, then the size is at least (roughly) $2^{n/4}$. If programs have length bounded by a constant c, then the size goes all the way up to $\frac{2^n}{q(n)}$, where q is a polynomial.

4 Instance complexity and CD complexity

This section contains a few observations about the connection between indistinguishability and instance complexity, on the one hand, and CD complexity, on the other.

First we observe that if either x or y has low distinguishing (CD) complexity, then the CD program may be used as a distinguisher for x and y.

Proposition 4.1 *For any x, y, and time bound t,*

$$Dist^t(x,y) \leq \min\{CD^t(x), CD^t(x,y)\}.$$

Next, we show that strings can be easily distinguished if there is an easy set which contains exactly one of x, y.

Proposition 4.2 *If $x \in A$ and $y \notin A$ then*

$$Dist(x, y) \leq ic(x : A) + ic(y : A) + O(1)$$

Proof: The proof is straightforward. Let p_0 and $p1$ be programs consistent with A of length ic$(x : A)$ and ic$(y : A)$, respectively, and such that $p_0(x), p_1(y) \neq \perp$. Patch the programs together to distinguish x and y as follows: on input z, simulate $p_0(z)$ and $p_1(z)$. Output the following value:

$$\begin{array}{ll} 0 & \text{if } p_0(z) = 0 \text{ or } p_1(z) = 0 \\ 1 & \text{if } p_0(z) = 1 \text{ or } p_1(z) = 1 \\ \perp & \text{otherwise} \end{array}$$

□

5 Nondeterministic communication complexity

The framework of distinguishability applies naturally and elegantly to the notion of nondeterministic communication complexity. First we give a brief overview of the model, then we provide a connection between the two notions. The connection takes the form of an upper bound and a lower bound on the communication complexity, and it is noteworthy that these bounds are tight in the sense that there are functions which meet the upper and lower bounds we provide.

5.1 Communication complexity model

First we present the model of nondeterministic communication complexity, introduced by Lipton and Sedgewick [LS81]. We recommend the survey paper by Lovász [Lo90] for more background on communication complexity.

The setting is the following: 2 players, A and B, want to evaluate a boolean function $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ on 2 arguments, x and y. A has x and B has y. They obtain help from a "divine source" in the form of a witness p drawn from a predetermined set of witnesses P. If $f(x, y) = 1$, then there is a p on which both A and B output 1. If not, there is no witness p that can cause both A and B to output 1. The goal is to minimize the length of p, which we may assume to be $\log |P|$. This length is the nondeterministic communication complexity for 1 instances of f, denoted $\kappa_1(f)$. The analogue for the 0-instances is denoted κ_0, where the witness set must have the property that a witness causes both parties to output 0 if and only if $f(x, y) = 0$.

We will associate with a communication problem f a matrix M_f whose rows are labeled by A's input, typically denoted by x, columns are labeled by B's input, typically y, and whose value in position x, y is the value $f(x, y)$.

5.2 A connection with distinguishability

It has long been known that the nondeterministic communication complexity has a precise combinatorial characterization in terms of homogeneous rectangle covers. A *rectangle* is a subset of the rows and columns of M_f. A *homogeneous rectangle* is a rectangle in which all the values $f(x, y)$ for which x is a row in the rectangle and y is a column in the rectangle share the same value. An all-1-rectangle is a homogeneous rectangle where all the values are 1, and all-0-rectangles are defined analogously. It is not hard to see that a family of all-0 rectangles which cover all the 0's in the communication matrix corresponds to a protocol for the 0 instances of f and vice versa. The same holds true for all-1 rectangle covers and protocols for 1-instances of f.

Distinguishers offer a dual viewpoint for nondeterministic communication complexity. We start by making a simple observation about how protocols can be viewed as distinguishers for inputs corresponding to 0 instances and those corresponding to 1 instances. Theorem 5.1 establishes that this connection goes both ways.

Claim 5.1 *If $f(x, y) \neq f(x, y')$ then either one of the 0- or 1-witnesses, together with B's strategy, serves as a distinguisher for y and y'.*

The triples x, y, y' can be thought of as obstacles to rectangles, and each obstacle must be countered with an appropriate distinguisher. In this sense, distinguishers provide us with a dual view of protocols.

Let P_f^A be a set of distinguishers which has the following property. For each x, and each pair y, y' such that $f(x, y) \neq f(x, y')$, there is a distinguisher $p \in P_f^A$ which distinguishes y from y'. Analogously, let P_f^B be a set of distinguishers for all the pairs x, x' for which there is a y such that $f(x, y) \neq f(x', y)$. Notice that if the length of the longest program in $P_f^A \bigcup P_f^B$ is d, then $|P_f^A \bigcup P_f^B|$ is at most 2^d.

Theorem 5.1 *Let P_f^A, P_f^B be as above, and denote by d the length of the longest distinguisher in $P_f^A \bigcup P_f^B$. Then*

$$d \leq \kappa_0, \kappa_1 \leq 2^{d+1}.$$

Proof: The first inequality follows directly from Claim 5.1.

To show the second inequality, we exhibit a protocol derived from $P_f^A \bigcup P_f^B$. We give the proof for 1 instances, as the proof for 0 instances is analogous.

Fix a pair of inputs x, y for which $f(x, y) = 1$. To convince A that $f(x, y) = 1$, it is enough to convince her that B's input is none of the inputs y' for which $f(x, y') = 0$. Likewise, to convince B that $f(x, y) = 1$, it is enough to establish that x is not any x' for which $f(x', y) = 0$.

The first part of the witness is an encoding of the distinguishers in P_f^A which differentiate y from all y' for which $f(x, y') = 0$. The second part of the witness is an encoding of the distinguishers in P_f^A which differentiate y from all y' for which $f(x, y') = 0$.

Both accept if they agree with all that has been put forth by the prover, namely, A checks x against all the distinguishers in the second part of the witness; B checks y against all the distinguishers in the first part of the witness. A also checks that the values of y not eliminated by the prover's distinguishers force f to 1; likewise B checks that all values of x not evaluated by the prover's distinguishers force f to 1.

Both parts of the witness can be encoded as a $\{0,1\}$ vector, where each position set to 1 indicate that the corresponding $p \in P_f^A$ or P_f^B must be checked against B's or A's input. Since there are at most 2^d programs of length $\leq d$, the witnesses have length $\leq 2^{d+1}$, which completes the proof of the theorem.

□

It is noteworthy that for the identity matrix (i.e., the communication matrix associated with the characteristic function $\chi_=$ of equality), it is known that $\kappa_0(\chi_=) = \log n$, and $\kappa_1(\chi_=) = n$. Clearly in this case $d = \log n$, so we have for this function that

$$\log n = d = \kappa_0(\chi_=)$$

and

$$\kappa_1(\chi_=) = 2^d = n.$$

In this sense our "dual" view of nondeterministic is optimal (up to a factor of 2 in the upper bound).

6 Further Research

There are several interesting open problems in the area communication complexity. Babai, Frankl and Simon [BFS86] defined the analogue of the polynomial-time hierarchy for communication complexity. Several separation results are known for the first and second levels of the communication complexity hierarchy, but it is as yet unknown if the entire hierarchy is proper. Can these questions be settled, or can we improve our understanding of them using indistinguishability?

Several other avenues warrant further study. Is there a clear connection to be made between the existence of large tangles and the density of non-proper complexity cores? Can the bounds on the size of these tangles be improved? Answers to any of these would provide useful insight to the question of the hardness of deciding instances of a language.

References

[BD87] Book, R. V., and D.-Z. Du, "The Existence and Density of Generalized Complexity Cores," *Journal of the ACM*, Vol. 34, No. 3, July 1987, pp. 718–730.

[BFS86] Babai, L., P. Frankl, and J. Simon, "Complexity classes in communication complexity theory", , 1986, pp. 337–347.

[BO94] Buhrman, H., and P. Orponen, "Random Strings Make Hard Instances", *Proceedings, Structures in Complexity Theory*, 1994, pp. 217–222

[FK95] Fortnow, L., and M. Kummer, "Resource-bounded instance complexity", *Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science*, LNCS, Springer, 1995.

[K95] Kummer, M., "Instance Complexity", *Proceedings of 10 Annual IEEE Conference on Structure in Complexity Theory*, 1995, pp. 111 – 124.

[LV93] Li, M., and Vitányi, P. (1993), "An Introduction to Kolmogorov Complexity and Applications", Springer-Verlag.

[LS81] Lipton, R., and R. Sedgewick, "Lower bounds for VLSI", *Proceeding of the 13 ACM Symposium on the Theory of Computing*, 1981, pp. 300 – 307.

[Lo90] Lovász, L., "Communication Complexity: A Survey", *Paths, Flows, and VLSI-layout*, Springer-Verlag 1990, pp. 235 – 266.

[Ly75] Lynch, N., "On reducibility to complex or sparse sets," *Journal of the ACM*, Vol. 22, 1975, pp. 341 – 345.

[OKSW94] Orponen, P., K.-I Ko, U. Schöning, and O. Watanabe, "Instance Complexity", *Journal of the ACM*, Jan. 1994.

[OS86] Orponen, P., and Schöning, U., "The density and complexity of polynomial cores for intractable sets", *Information and Control*, Vol. 70, 1986, pp. 54 – 68.