Indistinguishability

Sophie Laplante*
Université Paris-Sud (LRI)

John Rogers†
DePaul University

November 12, 1998

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, in time-bounded and time-unbounded settings. We show for every string length, there are large sets of strings that are pairwise indistinguishable. We compare indistinguishability with other (time-bounded) notions of Kolmogorov complexity: CD complexity and instance complexity.

If solving a problem requires distinguishing many hard-to-distinguish pairs, then the problem itself must also be hard. We illustrate this for three notions of hardness. We give a new characterization of regular languages computable languages using indistinguishability. We present a new indistinguishability-based technique for obtaining lower bounds in nondeterministic communication complexity, and give a corresponding upper bound which is optimal up to a factor of two.

Introduction

How hard is it to decide an instance of a language A? 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, FK96, 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.

To illustrate the expressiveness of indistinguishability, we give a simple characterization in terms of distinguishability of two classes: the regular languages, and the computable languages.

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.

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

*Laboratoire de Recherche en Informatique, Université Paris-Sud, Bâtiment 490, 91405 Orsay, France. Email: laplante@lri.fr Work done in part at the University of Chicago. Supported in part by NSF grant CCR 92-53582 and a Chateaubriand Postdoctoral Fellowship.

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

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 3.2 contains some basic results that provide motivation for studying indistinguishability. Section 4 links distinguishability with instance complexity and CD complexity (the time-bounded Kolmogorov complexity of recognizing strings.) Section 5 gives the characterizations of regular languages, and computable languages, in terms of indistinguishability. Finally, Section 6 establishes a relationship between communication complexity and distinguishability.

1 Preliminaries

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 computable 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 $P^{\leq k} = \{p_1, \ldots p_{N_k}\}$.

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 [LV97] for further background on Kolmogorov complexity.

In Definition 1.1, one may replace Φ by a clocked Turing machine which operates for at most a fixed number of steps $t(n)$ and then halts. Then the C^t complexity of a string x is the length of the shortest program which prints x after at most $t(|x|)$ steps. Clearly for any (time-constructible) t, and any x, $C^t(x) \geq C(x)$.

In the context of time-bounded computation, it makes sense to make the additional 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*

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

Since it is the case that recognizing a string is no harder than printing it, it is clear that for every time bound t and every x, $CD^t(x) \leq C^t(x)$.

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 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 (Orponen et al.) *Fix a universal Turing machine M with output symbols $\{0, 1, \perp\}$, which computes the partial computable function Φ. Fix also a computable set A.*
1. A Φ-program p is consistent *with A if for all x, when $\Phi(p, x) \neq \perp$ then $\Phi(p, x) = \chi_A(x)$.*
2. The t=-time-bounded 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 $\Phi(p, x) \neq \perp$.*

Time-bounded instance complexity, denoted $ic^t(x : A)$, is defined analogously.

2 Indistinguishability

We say two strings are distinguished by a program if the program, which can output 0, 1, or abstain from producing an output, outputs 1 on one string, and 0 on the other. We measure distinguishability in terms of the length of the shortest program which achieves this goal. In this section we give the formal definition of distinguishability, and we prove an invariance theorem.

2.1 Distinguishing programs

In order for our notion of indistinguishability to be robust, the programs we consider first are ones realized by a fixed partial computable 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 (t, k)-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 $\mathrm{Dist}^t_\Phi(x, y)$ *is the smallest k such that x, y are (Φ, t, k)-distinguishable. If $x = y$, then* $\mathrm{Dist}^t_\Phi(x, y) = \infty$.

If we allow the distinguishing programs to access an oracle A, then we write $\mathrm{Dist}^{t,A}_\Phi(x, y)$, and if we also allow a common auxiliary input string w, we write $\mathrm{Dist}^t_\Phi(x, y|w)$.

2.2 Invariance

Indistinguishability is a robust notion, in the sense that it is independent of the programming system Φ.

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

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

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 [LV97, Theorem 7.1.1, page 460]. 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 computable functions. Using the standard simulator U, we know that for each partial computable function $\Phi_i, U(\langle\langle i,\cdot\rangle,\cdot\rangle)$ 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 $\langle i,p\rangle$. □

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 computable function Φ.

3 Indistinguishable pairs

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.

In Kolmogorov complexity, strings which have Kolmogorov complexity at least their length are said to be incompressible. Analogously, indistinguishable pairs of strings are those which require at least $\log(n)$-length programs to be distinguished.

3.1 Existence of indistinguishable pairs

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

Theorem 3.1 *$\forall n, k, 0 < k < \log(n)$, there are at least $2^{2n-2^{k+1}} - 2^{n-1}$ pairs $x \neq y, |x| = |y| = n$ such that $\mathrm{Dist}^t_\Phi(x,y) \geq k$.*

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.

A counting argument completes 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 constant fraction of the total number of pairs of distinct strings of length n: $2^{2n} - 2^n$.

3.2 Large sets of pairwise indistinguishable strings

From the proof of Theorem 3.1, 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 $\forall x, y \in T, \mathrm{Dist}^t_\Phi(x, y) > k$. We say T is a (t, k)-tangle when Φ is a fixed universal partial computable 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 theorem that any string of high Kolmogorov complexity must be involved in a large tangle.

Theorem 3.2 *$\forall n, k, 0 < k < \log(n), |x| = n$, 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: Observe that the set of inputs with signatures compatible with a given string is enumerable. 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 in the enumeration of 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 Theorem 3.1, $P^{\leq k}$ 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. □

Tangles associated with Kolmogorov-incompressible strings are surprisingly large. 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.

We can get a similar result for time-bounded computation using a result of Buhrman and Fortnow [BF97]. Their theorem allows us to bound the CD complexity of instances in sets of bounded size.

Theorem 3.3 (Buhrman-Fortnow) *For any set A (accessible as an oracle), there is a polynomial p and a constant c such that for all strings $x \in A \cap \Sigma^n$,*

$$CD^{p,A}(x) \leq 2\log(|A \cap \Sigma^n|) + c\log n.$$

Using their result we obtain the following analogue of Theorem reftangle-lemma for polynomial-time CD complexity.

Theorem 3.4 *For every $n, k, 0 < k < \log(n), |x| = n$, and polynomial q, there is a polynomial p and a (q,k)-tangle T_x including x with*

$$|T_x| \geq 2^{\frac{CD^{2^k}p_{(x)}}{2} - 2^k - \log(k) - O(\log(n))}.$$

The proof is analogous to that of Theorem 3.2, but instead of including x's index into the tangle, we include its distinguishing program as given by Theorem 3.3.

4 Instance complexity, CD complexity, and language compression

There are some natural relationships between distinguishability and other notions of (time-bounded) Kolmogorov complexity. We consider instance complexity, which measures the length of the shortest program that computes a specific instance of a problem. We also consider CD complexity (the complexity of recognizing strings) and instance complexity (the length of the shortest program that can compute a specific instance of a problem.) An important problem in CD complexity is the language compression problem: given a set of strings of equal length, is there a set of short polynomial-time programs, each of which recognizes exactly one of the strings in the set? The results in this section illustrate the relationship between distinguishability and these notions.

4.1 Instance complexity

The following proposition states that the strings x, y can be easily distinguished if there is a set which is easy to compute on those instances and which contains exactly one of x, y.

Proposition 4.1 *If $(x,y) \in A \times \overline{A}$, and for any time bound t, then*

$$\mathrm{Dist}^{O(t)}(x,y) \leq \mathrm{ic}^t(x : A) + \mathrm{ic}^t(y : A) + O(1)$$

Proof: Let p_0 and p_1 be t-time bounded programs consistent with A of length $\mathrm{ic}^t(x : A)$ and $\mathrm{ic}^t(y : A)$, 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:

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

The first two cases are mutually exclusive because p_0 and p_1 are consistent with A, and the resulting program correctly distinguishes x from y. □

The instance complexity relative to a set A can also be bounded by distinguishers of instances in and out of the set, as we show below.

Proposition 4.2 *For every set A, time bound t, and instance z,*

$$\mathrm{ic}^{2^d t}(z : A) \leq 2^{d+1},$$

where $d = Max_{(x,y) \in A \times \overline{A}}\{\mathrm{Dist}^t(x,y)\}$.

Proof: To solve instance z correctly, run z on all distinguishing programs of length no more than d. Then compare all of these answers to a hard-coded "cheat-sheet" containing all the correct answers for z. If the answers all agree, conclude z is in the language; otherwise output $\perp$. This program must halt and output the correct answer on z. On all other inputs z', if the program does not output $\perp$, then z' must be in A by maximality of d, so the program is consistent with A as required. □

The previous proposition states that distinguishing all in-out pairs with $\log(l)$ length distinguishers implies l instance complexity. However, it is not the case that distinguishing these pairs requires only the logarithm of instance complexity. To wit:

Proposition 4.3 *Fix any l. Then there is a set A such that the instance complexity of an instance of A is at most l, but there is a pair of instances x, y such that $x \in A$ and $y \notin A$, and $\mathrm{Dist}(x, y)$ is at least l.*

Proof: Consider two indistinguishable instances of length 2^l. Take the shortest program p (of length l) that distinguishes x and y. Let $A = \{z | p(z) \text{ outputs } 1\}$. Clearly for any z (time-bounded, but arbitrary time bound) $ic(z : A)$ is at most l, since p basically computes A. However $x \in A$ and $y \notin A$ (or vice versa) yet there is no distinguisher of length $< l$ that distinguishes them. □

4.2 CD complexity and language compression

Distinguishing (CD) complexity is a natural upper bound for distinguishability. If either x or y has low CD complexity, then the CD program may be used as a distinguisher for x and y.

Proposition 4.4 *For any x, y, and time bound t,*

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

In language compression problem, we are given a set of d strings of equal length, and the goal is to show that is there a set of short polynomial-time programs which can access the set as an oracle, and each of which recognizes exactly one of the strings in the set. This is known to have many applications in complexity, such as the Valiant-Vazirani randomized reduction from boolean formulas to formulas which have at most one satisfying assignment, enumerating satisfying assignments, and showing that BPP is in the polynomial hierarchy. (These applications and more can be found in a paper of Buhrman and Fortnow [BF97].) Clearly, the best upper bound one can hope for is to have programs of length $\log d$, since d programs are required for sets of size d. (In fact, $\log d + \log n$ length is required to account for the string length n.)

Theorem 3.3 gives a construction for programs of length $2 \log d + O(\log n)$, and this is the best bound known in the deterministic setting. It immediately follows from Theorem 3.3 that for any x, y in A, $\mathrm{Dist}^{p,A}(x, y) \leq 2 \log(|A \cap \Sigma^n|) + O(\log n)$.

The proof of Theorem 3.3 (attributed to John Tromp [BF97]) hinges on the following lemma, which allows us to obtain a much better bound on distinguishing complexity for all pairs in the set.

Lemma 4.1 [BF97] *Let $A \subseteq \{0,1\}^n$ with $|A| = d$. Then there is a prime $p \leq 2d^2 n$ such that for all $x, y \in A, x \neq y, x \not\equiv y \bmod p$.*

Corollary 4.1 *Let $A \subseteq \{0,1\}^n$ with $|A| = d$. Then there is a string w of length $2 \log d + \log n$ such that for any $x, y \in A, x \neq y, \mathrm{Dist}^{p,A}(x, y|w) \leq 2 \log \log d + \log \log n$, where p is a fixed polynomial.*

Proof: It suffices to encode an index into the position where $x \bmod p$ differs from $y \bmod p$. □

We note that in Buhrman and Fortnow [BF97] the proof that $BPP \in \Sigma_2^p \cap \Pi_2^p$ actually uses Lemma 4.1 instead of Theorem 3.3. The natural Kolmogorov complexity property needed in the proof is the existence of small distinguishing programs for all pairs of strings in the set of accepting paths for a BPP machine.

Short distinguishing programs for all pairs in a set A also give a very general bound on the language compression problem.

Proposition 4.5 *Let $A \subseteq \{0,1\}^n$ with $|A| = d$. If for any $x, y \in A, x \neq y, \mathrm{Dist}^{p,A}(x, y|w) \leq k$, then for all $z \in A, CD^{2^k p,A}(z) \leq 2^k$.*

Proof: A tag compatible with the output of each program of length k on input x will uniquely identify x, since otherwise, for some y, no program distinguishes x and y. □

5 The expressiveness of indistinguishability

Indistinguishability can be used to characterize two important classes of languages: the regular languages, and the computable languages. These characterizations are simple and illustrate the expressiveness of indistinguishability.

5.1 A characterization of regular languages

Kolmogorov complexity has been used to give a characterization of regular languages (KC-regularity, Li and Vitányi [LV97, LV95, Section 6.8].)

Theorem 5.1 [LV95] *Let $L \subseteq \{0,1\}^*$. Then L is regular if and only if there exists a constant c depending only on L such that*

$$\forall x \in \{0,1\}^*, \forall n, C(\chi_{1:n}) \leq C(n) + c.$$

Using indistinguishability, we characterize the regular languages in the following way.

Theorem 5.2 *Let $L \subseteq \{0,1\}^*$. Then L is regular if and only if there is a constant c depending only on L such that for any pair x, y for which there exists a suffix z for which $xz \in L$ if and only if $yz \notin L$, $\mathrm{Dist}^t(x,y) \leq c$, where t is a linear time bound on the running time.*

Proof: For the implication, the states of the automata serve as the distinguishers. For the converse, let $p_1 ... p_k$ be the distinguishing programs of length bounded by c as above. Associate a state with each outcome of $p_1 ... p_k$ (2^k states) and label it with any string which has reached this state from the start state. The state s is a final state if the label associated with it is in the language.

A transition is made from state s to s' if the string at state s concatenated with the current symbol has as its outcome when run on $p_1 ... p_k$, the outcome at state s'. □

Example 5.1 *The language $L = \{xx | x \in \{0,1\}^*\}$ is not regular.*

Proof: Assume, for a contradiction, that L is regular. Take two indistinguishable strings x, y of length m. Since $xx \in L$ and $xy \notin L$, by Theorem 5.1 there is a distinguisher of length at most c to distinguish x and y. If we choose $m = c + 1$ we have a contradiction. □

5.2 A characterization of computable languages

The computable languages can also be characterized using an indistinguishability property, as we see now.

Theorem 5.3 *A is computable if and only if there is a constant c depending only on A such that for every pair $(x,y) \in A \times \overline{A}$, $\mathrm{Dist}(x,y) \leq c$.*

Proof: For the implication, a single program for A can distinguish any pair of strings x, y as required.

For the converse, we have a constant number of programs to consider, one of which is guaranteed to tell x, y apart if $L(x) \neq L(y)$. Assuming first that all the distinguishers will halt, we can simulate all the distinguishers on the given input, and look up in a table what to answer given the list of outcomes of all the distinguishers. (If two strings are not distinguished, then they are surely both in, or both out of the language.)

However we do need to handle the possibility that one or more distinguishers do not halt on the given input. Instead of simulating all distinguishers on the input, we dovetail.

Assume some outcome is missing from the list of outcomes, i.e., some distinguisher has not yet halted. Either the outcome is not necessary to determine whether or not the input is in A: regardless of the outcome, if any, it is either in, or out of A. On the other hand, if it is necessary, that is, an outcome of 0 places it in A and 1 places it in $\overline{A}$, or vice versa, then these two outcomes must be distinguishable. Therefore the program will halt eventually. This argument generalizes easily to the case where several distinguishers fail to halt on an input. □

6 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 both the upper and lower bounds we provide.

6.1 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] and the book by Kushilevitz and Nisan [KN97] for more background on communication complexity.

The setting is the following: two players, A and B, want to evaluate a boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ on two 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 $\kappa_0(f)$, 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 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 $f(x,y)$.

6.2 Distinguishability for nondeterministic communication complexity

The nondeterministic communication complexity of a function 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 observing that protocols can be viewed as distinguishers for inputs corresponding to 0 instances and those corresponding to 1 instances. Theorem 6.1 establishes that this connection goes both ways.

Proposition 6.1 *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. By looking at the obstacles to rectangles instead of the rectangles themselves, 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' given the communication matrix M_f. 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 $P_f^A \cup P_f^B$ is d, then $|P_f^A \cup P_f^B|$ is at most 2^d.

Theorem 6.1 *Let P_f^A, P_f^B be as above, and denote by d the length of the longest distinguisher in $P_f^A \cup P_f^B$. Then*

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

Proof: The first inequality follows directly from Proposition 6.1.

To show the second inequality, we exhibit a protocol derived from $P_f^A \cup P_f^B$. We give the proof for 1 instances, and 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 $\leq 2^{d+1}$. □

Note that this also yields and upper bound for deterministic communication complexity.

Example 6.1 *For the equality function, $\kappa_0(\chi_=) \geq \log n$.*

Proof: Let x, y be indistinguishable strings of length n. Then $d = \log n$, and $\kappa_0(\chi_=) \geq d = \log n$. □

Indeed, $\kappa_0(\chi_=) = \log n$, and $\kappa_1(\chi_=) = n$, so Theorem 6.1 is optimal (up to a factor of two in the upper bound).

7 Further Research

There are several interesting open problems in the area of 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.

Acknowledgments

We would like to thank Lance Fortnow, Stuart Kurtz, Janos Simon and Michael O'Donnell for comments and suggestions on this work. We thank Amber Settle for many productive discussions, and Marcus Schaefer and Dieter van Melkebeek for comments on an earlier draft.

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", *Proceedings of the 27th Symposium on Foundations of Computer Science*, 1986, pp. 337 – 347.

[BF97] H. Buhrman and L. Fortnow. "Resource-bounded Kolmogorov complexity revisited," *Proceedings of the 14th Symposium on Theoretical Aspects of Computer Science*, LNCS, Springer, 1997.

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

[FK96] L. Fortnow and M. Kummer. Resource-bounded instance complexity. *Theoretical Computer Science A*, 161:123–140, 1996.

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

[KN97] Kushilevitz, E., and Nisan, N., "Communication Complexity", Cambridge University Press, 1997.

[LV97] Li, M., and Vitányi, P., "An Introduction to Kolmogorov Complexity and Its Applications", Second Edition, Springer-Verlag, 1997.

[LV95] Li, M., and Vitányi, P., "A new approach to formal language theory by Kolmogorov complexity", *SIAM Journal on Computing*, 24(2):398–410, 1995.

[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 Computation*, Vol. 70, 1986, pp. 54 – 68.