Instance Complexity

PEKKA ORPONEN
University of Helsinki, Finland
KER–I KO
State University of New York at Stony Brook, New York, USA
UWE SCHÖNING
Universität Ulm, Germany
OSAMU WATANABE
Tokyo Institute of Technology, Japan

Abstract

We introduce a measure for the computational complexity of individual instances of a decision problem and study some of its properties. The instance complexity of a string x with respect to a set A and time bound t, $\mathrm{ic}^t(x : A)$, is defined as the size of the smallest special-case program for A that runs in time t, decides x correctly, and makes no mistakes on other strings ("don't know" answers are permitted). We prove that a set A is in P *if and only if there exist a polynomial t and a constant c such that $\mathrm{ic}^t(x : A) \leq c$ for all x; on the other hand, if A is* NP*-hard and* P $\neq$ NP*, then for all polynomials t and constants c, $\mathrm{ic}^t(x : A) > c \log |x|$ for infinitely many x. Observing that $K^t(x)$, the t-bounded Kolmogorov complexity of x, is roughly an upper bound on $\mathrm{ic}^t(x : A)$, we proceed to investigate the existence of individually hard problem instances, i.e. strings whose instance complexity is close to their Kolmogorov complexity. We prove that if $t(n) \geq n$ is a time constructible function and A is a recursive set not in $DTIME(t)$, there then exist a constant c and infinitely many x such that $\mathrm{ic}^t(x : A) \geq K^{t'}(x) - c$, for some time bound $t'(n)$ dependent on the complexity of recognizing A. Under the stronger assumptions that the set A is* NP*-hard and* DEXT $\neq$ NEXT*, we prove that for any polynomial t there exist a polynomial t' and a constant c such that for infinitely many x, $\mathrm{ic}^t(x : A) \geq K^{t'}(x) - c$. If A is* DEXT*-hard, then the same result holds unconditionally. We also prove that there is a set $A \in$* DEXT *such that for some constant c and all x, $\mathrm{ic}^{\exp}(x : A) \geq K^{\exp'}(x) - 2 \log K^{\exp'}(x) - c$, where $\exp(n) = 2^n$ and $\exp'(n) = cn2^{2n} + c$.*

[0]Preliminary versions of parts of this work have appeared under the titles "What is a hard instance of a computational problem?" in *Proceedings of the Conference on Structure in Complexity Theory* (Berkeley, Ca., June 1986), and "On the instance complexity of NP-hard problems" in *Proceedings of the 5th Annual Conference on Structure in Complexity Theory* (Barcelona, July 1990). These Proceedings have been published by Springer-Verlag, Berlin, and IEEE, New York, respectively.

Authors' present addresses: P. Orponen, Department of Computer Science, University of Helsinki, SF–00510 Helsinki, Finland. K. Ko, Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794, USA. Uwe Schöning, Abteilung Theoretische Informatik, Universität Ulm, D–7900 Ulm, Germany. Osamu Watanabe, Department of Computer Science, Tokyo Institute of Technology, Tokyo 152, Japan. The research of P. Orponen was supported by the Academy of Finland, and the research of K. Ko in part by NSF grant CCR–8801575.

1 Introduction

One can try to understand the computational intractability of decision problems from two, perhaps complementary, points of view. A "distributional" view holds that the positive and negative instances of a difficult problem are distributed in some very irregular manner, and feasible algorithms can only determine simple distributions. This is the customary mode of thought in complexity theory, where the asymptotic behaviour of algorithms is emphasized. An alternative view is suggested by the intuition that also individual problem instances can be in some sense inherently hard, i.e., hard independent of any particular algorithm used to decide the problem. Such ideas of "instance complexity" have been discussed by, for instance, Hartmanis in [11].

One proposed approach to studying this dichotomy has been via the notion of complexity cores, introduced by Lynch in [21]. Let us consider decision problems encoded as sets of strings. A *(polynomial) complexity core* for a set A is a set C such that for every algorithm M that decides A, and every polynomial t, M needs more than $t(|x|)$ time on all but finitely many x in C. Thus, one could plausibly interpret a complexity core as an inherently hard collection of problem instances. It is known that any recursive set not in P has an infinite polynomial complexity core [21], and that if P $\neq$ NP, then NP-complete sets have cores whose density, i.e., the number of strings of each length in the core, is not bounded by any polynomial function [26]. (For an overview of recent work on complexity cores, see [6]).

Nevertheless, the notion of a complexity core does not seem to lead to any useful formulations of the idea of single instance complexity. Because of the "all but finitely many" provision in the definition, any finite variation to a core is still a core, and the provision cannot be removed because any finite set of instances can be decided in constant time by table look-up. This possibility of patching algorithms by finite tables is the main difficulty in formulating a satisfactory notion of single instance complexity.

However, the difficulty can be overcome by taking also the *sizes* of algorithms into account. Here we propose the following approach. Consider the class of Turing machines that on each input can output either 1 (accept), 0 (reject), or $\perp$ (don't know). A machine M in this class is *consistent with* a set A if for all inputs x such that $M(x) \neq \perp$, $M(x) = 1$ if and only if $x \in A$. Given a set A and a time bound t, the *t-bounded instance complexity of x with respect to A* is defined as

$$\begin{aligned}\mathrm{ic}^t(x : A) = \min\{|M| : \ & M \text{ is consistent with } A, \\ & \mathrm{time}_M(y) \leq t(|y|) \text{ for all } y, \text{ and } M(x) \neq \perp\}.\end{aligned}$$

Actually, the measure $|M|$ used here, "the size of Turing machine M", is not a well-defined notion, and the definition should really be framed in terms of *programs* to some fixed, sufficiently efficient universal machine. In the body of the paper we will use the correct definition, but the above suffices for purposes of discussion.

Technically, our definition is obviously inspired by the notion of Kolmogorov complexity [10, 18, 19], which provides a measure for the complexity of an individual string. Recall that the *t-bounded Kolmogorov complexity of a string x* is defined (roughly) as

$$K^t(x) = \min\{|M| : \mathrm{time}_M(\lambda) \leq t(|x|), \text{ and } M(\lambda) = x\},$$

where λ denotes the empty string. There is also an interesting variant of this, introduced by Sipser in [29]:

$$KD^t(x) = \min\{|M| : \mathrm{time}_M(y) \leq t(|y|) \text{ for all } y, \text{ and } M(y) = 1 \text{ if and only if } y = x\}.$$

It is easy to see that $KD^t(x) \leq K^t(x)$ for all t and x.

Although formally similar, the issues addressed by the instance complexity and Kolmogorov measures are rather different. The Kolmogorov measures are concerned with the combinatorial complexity of a string x as such, whereas the ic measure indicates the complexity of determining whether a given string x has a certain property A — although Sipser's KD measure can be viewed also as a special case of instance complexity, because $KD^t(x) = \mathrm{ic}^t(x : \{x\})$. An early variant of Kolmogorov complexity that is somehow close in spirit to instance complexity is Loveland's *uniform complexity* $K(A; x)$ [20]. (A time-bounded version of this is discussed in [15].) In our notation, Loveland's definition can be formulated as:

$$K(A;x) = \min\{|M| : \text{ for all } y \leq x,\ M(y) \neq \bot \text{ and } M(y) = 1 \text{ iff } y \in A\}.$$

In the following, we first in Section 2 formulate the proper definitions of instance complexity and related notions. Then, in Section 3, we map out some elementary properties of the new measure. For instance, we show that a set A is in P if and only if there exist a polynomial t and a constant c such that $\mathrm{ic}^t(x : A) \leq c$ holds for all x. We also give a simple characterization of complexity cores in terms of instance complexity, and consider the behaviour of the ic measure under polynomial time reductions.

In Section 4 we study the very interesting class of sets

$$\begin{aligned}\mathrm{IC}[\log, \mathrm{poly}] = \{A :\ & \text{for some constant } c \text{ and polynomial } t, \\ & \mathrm{ic}^t(x : A) \leq c \log|x| + c \text{ for all } x\}.\end{aligned}$$

We show that for any polynomially self-reducible [23, 28] set A, and also for any set that is $\leq^p_{btt}$-hard for NP, $A \in \mathrm{IC}[\log, \mathrm{poly}]$ if and only if $A \in \mathrm{P}$. We also relate the new class to the advice complexity classes P/log and P/poly defined by Karp and Lipton [14] by showing that

$$\mathrm{P/log} \subsetneq \mathrm{IC}[\log, \mathrm{poly}] \subsetneq \mathrm{P/poly}.$$

Thus, our result about the instance complexity of self-reducible or NP-hard sets is a provable improvement to Karp's and Lipton's result that SAT $\in$ P/log if and only if P = NP.

In the most fundamental Section 5, we study the existence of intrinsically hard problem instances. Note that Kolmogorov complexity provides an upper bound for instance complexity, because size $KD^t(x)$ is sufficient to recognize, by table look-up, the string x. Thus, an instance x may be considered to be "intrinsically hard", with respect to a problem A and time bound t, if the value of $\mathrm{ic}^t(x : A)$ is close to $KD^t(x)$. Intuitively, this means that no method for deciding any subproblem of A in time t can do substantially better on x than simply treat it as an individual special case, and store a description of x in a table. Based on this technical formulation of the intuitive idea of instance hardness, we propose the following very strong "instance complexity conjecture": that for all appropriate time bounds t, if a set A is not in DTIME(t), there then exist a constant c and infinitely many x such that $\mathrm{ic}^t(x : A) \geq KD^t(x) - c$. (Technically, we may have to allow for a $\log t$ factor of slack between the time bounds for the instance and Kolmogorov complexities — cf. Theorem 2.1 in the next section.)

As partial support for the conjecture, we prove in Section 5 that if $t(n) \geq n$ is a time constructible function and A is a recursive set not in DTIME(t), there then exist a constant c and infinitely many x such that $\mathrm{ic}^t(x : A) \geq KD^{t'}(x) - c$, where the time bound $t'(n)$ depends on the complexity of recognizing A. (We actually prove the result in the slightly stronger form that for infinitely many x, $\mathrm{ic}^t(x : A) \geq K^{t'}(x) - c$.) The dependency of the time bound $t'(n)$ on the complexity of A is unfortunate, but with a more involved construction we are able to essentially remove it in many interesting special cases. Namely, we prove that if DEXT $\neq$ NEXT, then for any set A that is $\leq^p_{1-tt}$-hard for NP, and for any polynomial t there exist another polynomial t' and a constant c such that for infinitely many x, $\mathrm{ic}^t(x : A) \geq K^{t'}(x) - c$. For DEXT-hard sets A, the same result holds even without the assumption DEXT $\neq$ NEXT. Yet another result in Section 5 shows that there exists a set $A \in$ DEXT such that for some constant c and *all* x,

$$\mathrm{ic}^{\exp}(x : A) \geq K^{\exp'}(x) - 2 \log K^{\exp'}(x) - c,$$

where $\exp(n) = 2^n$ and $\exp'(n) = cn2^n + c$. As a corollary, we obtain that all DEXT-complete sets have exponentially dense sets of instances with a similar property.

Section 6 provides a brief summary and suggests some further research directions.

2 Preliminaries

We consider decision problems coded as sets of strings over the alphabet $\Sigma = \{0, 1\}$. The length of a string $x \in \Sigma^*$ is denoted $|x|$; λ denotes the empty string. We define a pairing function on strings as follows: given strings x, y, let the binary representation of $|x|$, without leading zeros, be $b_1 \ldots b_k$; then $\langle x, y\rangle = b_1 b_1 \ldots b_k b_k 10xy$. Clearly both the pairing function, and the associated projection functions can be computed by multitape Turing machines in linear time, and for all x and y, $|\langle x, y\rangle| \leq |x| + |y| + 2\log|x| + 4$[1].

An *interpreter* is a deterministic Turing machine M with two input tapes (a "program" tape and a "real input" tape) and an arbitrary number of work tapes, one of which is designated as the output tape. The input and output tape alphabets of M are Σ. M accepts its input if at the end of a computation the output tape contains the string "1", rejects if the output tape contains a "0", and is *undecided* if the computation does not halt or if at its end the output tape contains something else — we denote both of these outcomes generically as "$\perp$". The partial mapping from $\Sigma^* \times \Sigma^*$ to Σ^* computed by M is denoted $M(p, x)$, and the time requirement of M on input (p, x) is denoted $\mathrm{time}_M(p, x)$. The partial mapping computed by M on a fixed program string p from Σ^* to Σ^* is denoted $f^M_p(x)$. Program p is *total* (w.r.t. interpreter M) if $f^M_p(x) \neq \perp$ for all x.

For a set of strings A, we use the notation $A(x)$ for the characteristic function of A, i.e., $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$. For $b \in \{0, 1\}$, we denote $M(p, x) \simeq b$ (read $M(p, x)$ *is consistent with* b) if $M(p, x) = b$ or $M(p, x) = \perp$. In particular, for a set A and strings p, x, $M(p, x) \simeq A(x)$ means that if $M(p, x) \neq \perp$, then $M(p, x) = 1$ if and only if $x \in A$.

Definition 2.1 Let M be an interpreter, A a set of strings, and t a function on the natural numbers. A string p is an *(M, t)-program for A* if for all strings y, $\mathrm{time}_M(p, y) \leq t(|y|)$ and $M(p, y) \simeq A(y)$. Program p *decides* x if $M(p, x) \neq \perp$. The *t-bounded instance complexity* of a string x *with respect to A using M* is defined as

$$\mathrm{ic}^t_M(x : A) = \min\{|p| : p \text{ is an } (M, t)\text{-program for } A \text{ deciding } x\}.$$

If no (M, t)-program for A decides x, $\mathrm{ic}^t_M(x : A)$ is taken to be infinite.

[1]All the log's in this paper are to base 2. For the purposes of this paper, it is convenient to define $\log 0 = 0$.

Definition 2.2 Let M be an interpreter, t a function on the natural numbers, and x a string. A string p is an *(M,t)-program for producing x* if $\mathrm{time}_M(p,\lambda) \leq t(|x|)$ and $f_p^M(\lambda) = x$. The *t-bounded Kolmogorov complexity of x using M* is defined as

$$K_M^t(x) = \min\{|p| : p \text{ is an } (M,t)\text{-program for producing } x\}^2.$$

If no (M,t)-program produces x, $K_M^t(x)$ is taken to be infinite.

The fundamental property of these notions is that they can actually be defined very robustly by means of a universal interpreter.

Theorem 2.1 (Invariance) *There exists an interpreter U such that corresponding to any other interpreter M there is a constant c, such that for all sets A, time bounds t and strings x,*

$$\begin{aligned} \mathrm{ic}_U^{t'}(x:A) &\leq \mathrm{ic}_M^t(x:A) + c, \\ K_U^{t'}(x) &\leq K_M^t(x) + c, \end{aligned}$$

where $t'(n) = ct(n)\log t(n) + c$.

Proof. See [10, 18, 19]; this is the standard result on the invariance of time-bounded Kolmogorov complexity, using the efficient Hennie–Stearns simulation (see [12, Sec. 12]) of multitape machines by two-tape machines. □

Because the complexities obtained using U essentially minorize the complexities obtained using any other interpreter, we define absolutely the *t-bounded instance complexity of x with respect to A* as $\mathrm{ic}^t(x : A) = \mathrm{ic}_U^t(x : A)$, and the *$t$-bounded Kolmogorov complexity of x* as $K^t(x) = K_U^t(x)$. We then call a (U,t)-program p simply a t-program, and denote $f_p(x) = f_p^U(x)$, and $\mathrm{time}_p(x) = \mathrm{time}_U(p,x)$.

We define the deterministic time complexity classes with respect to programs on U, not arbitrary Turing machines. This results in slightly nonstandard definitions for the more sensitive classes, but has no effect on classes such as P, DEXT, etc. Let us denote $L_p = f_p^{-1}(1)$. Then

$$\begin{aligned} \mathrm{DTIME}(t(n)) &= \{L_p : \mathrm{time}_p(x) \leq ct(|x|) \text{ for some constant } c\}, \\ \mathrm{P} &= \bigcup_{c>0} \mathrm{DTIME}(n^c + c), \\ \mathrm{DEXT} &= \bigcup_{c>0} \mathrm{DTIME}(2^{cn} + c). \end{aligned}$$

In order to guarantee that the classes DTIME(t), as defined above, are closed under the Boolean operations and simple transformations on Σ^*, we assume w.l.o.g. that the programming system determined by U is *efficiently closed* under Boolean operations and composition. By this we mean that there exists a constant γ such that for any pair of everywhere halting programs p, q there exist programs $p \cup q, \neg p$, and $p \circ q$ such that

$$f_{p\cup q}(x) = \begin{cases} 1, & \text{if } f_p(x) = 1, \text{ or } f_p(x) = \perp \text{ and } f_q(x) = 1, \\ 0, & \text{if } f_p(x) = 0, \text{ or } f_p(x) = \perp \text{ and } f_q(x) = 0, \\ \perp, & \text{otherwise;} \end{cases}$$

[2]Only this version of Kolmogorov complexity is used in the body of the paper.

$$\begin{aligned} f_{\neg p}(x) &= \begin{cases} 1, & \text{if } f_p(x) = 0, \\ 0, & \text{if } f_p(x) = 1, \\ \perp, & \text{otherwise,} \end{cases} \\ f_{p \circ q}(x) &= f_p(f_q(x)), \end{aligned}$$

and

$$\begin{aligned} |p \cup q| &\leq |p| + |q| + 2\log|p| + \gamma, \\ |\neg p| &\leq |p| + 2\log|p| + \gamma, \\ |p \circ q| &\leq |p| + |q| + 2\log|p| + \gamma, \end{aligned}$$

$$\begin{aligned} \mathrm{time}_{p \cup q}(x) &\leq \begin{cases} \mathrm{time}_p(x) + \gamma|p \cup q|, & \text{if } f_p(x) \neq \perp, \\ \mathrm{time}_p(x) + \mathrm{time}_q(x) + \gamma|p \cup q|, & \text{if } f_p(x) = \perp, \end{cases} \\ \mathrm{time}_{\neg p}(x) &\leq \mathrm{time}_p(x) + \gamma|\neg p|, \\ \mathrm{time}_{p \circ q}(x) &\leq \mathrm{time}_q(x) + \mathrm{time}_p(f_q(x)) + \gamma|p \circ q|. \end{aligned}$$

Such structure can be imposed on the programming system by using a pairing function similar to the one described above to encode pairs of elementary programs, together with some information as to how the pair is to be interpreted. The operations can naturally be iterated; in particular, we define

$$p_1 \cup p_2 \ldots \cup p_k = p_1 \cup (p_2 \cup \ldots (p_{k-1} \cup p_k) \ldots).$$

An important property of this iterated union is that for any fixed set of programs $p_1, \ldots, p_k$,

$$\mathrm{time}_{p_1 \cup p_2 \ldots \cup p_k}(x) = O(\max_{1 \leq i \leq k} \mathrm{time}_{p_i}(x)).$$

3 Elementary Properties

Using table look-up, the Kolmogorov complexity of a string is easily seen to be an upper bound on its instance complexity with respect to any set.

Proposition 3.1 *For any time constructible function t, there exists a constant c such that for any set A and string x,*

$$\mathrm{ic}^{t'}(x : A) \leq K^t(x) + c,$$

where $t'(n) = ct(n)\log t(n) + c$.

Proof. Given a time constructible t, consider an interpreter M that works as follows: on input $(\langle b, p\rangle, y)$, where $b \in \Sigma, p \in \Sigma^*, y \in \Sigma^*$, M simulates $U(p, \lambda)$ for $t(|y|)$ steps. If $U(p, \lambda)$ halts in this time with output y, M outputs b and halts, otherwise M outputs λ and halts. Clearly there is a constant d such that for any b, p, and y, M halts in time bounded by $t(|y|) + d$. Let then A be any set, and x a string. Let $b = A(x)$, and let p be a minimal length t-program for producing x. Then $\langle b, p\rangle$ is an $(M, t + d)$-program for A deciding x, and so

$$\mathrm{ic}_M^{t+d}(x : A) \leq |\langle b, p\rangle| = |p| + 5 = K^t(x) + 5.$$

By invariance (Theorem 2.1), then, there is a constant c, independent of A and x, such that

$$\mathrm{ic}^{t'}(x : A) \le K^t(x) + c,$$

where $t'(n) = ct(n)\log t(n) + c$. □

The notion of instance complexity allows for very simple and elegant characterizations of many fundamental complexity-theoretic properties, as the following examples show.

Proposition 3.2 *A set A is in* P *if and only if there exist a polynomial t and a constant c such that for all x,* $\mathrm{ic}^t(x : A) \le c$.

Proof. Assume first that A is in P. Let t' be a polynomial and p a t'-program such that for all x, $x \in A$ if and only if $U(p, x) = 1$. Let $q = \chi \circ p$, where χ is a constant-time program such that $f_\chi(1) = 1, f_\chi(y) = 0$ for $y \neq 1$. Then $U(q, x) = A(x)$ for all x, and there is a constant d such that $|q| \le |p| + d$ and $\mathrm{time}_q(x) \le t'(|x|) + 2\log|p| + d$ for all x. Hence, denoting $c = |p| + d$, $t(n) = t'(n) + 2\log|p| + d$, we obtain that $\mathrm{ic}^t(x : A) \le |q| \le c$ for all x.

Conversely, assume that there are a polynomial t and a constant c such that for all x, $\mathrm{ic}^t(x : A) \le c$. Then among the finitely many programs of size at most c there is a set, say $p_1, \ldots, p_k$, of t-programs for A, such that for every x, $U(p_i, x) \neq \perp$ for at least one $i \in \{1, \ldots, k\}$. But then $p = p_1 \cup \ldots \cup p_k$ is a total $O(t)$-program for A, witnessing that $A \in$ P. □

Definition 3.1 Let A be a recursive set. A set C is a *polynomial complexity core for A* if C is infinite, and for every total program p for A and polynomial t, $\mathrm{time}_p(x) > t(|x|)$ for almost all x in C (i.e., for all but finitely many x in C). A set A is *p-immune* if it is a polynomial core for itself, and *bi-immune* if both A and $\bar{A}$ are cores for A.

The notion of a polynomial complexity core was defined by Lynch [21] and further studied by various authors in, e.g., [8, 26]. The idea of immunity was transported from its original recursion theoretic setting (cf. [27, §8.2]) to complexity theory by Flajolet and Steyaert in [9], although the idea was already anticipated by Chaitin in [7]. Bi-immunity was introduced by Balcázar and Schöning in [3]. We obtain the following characterizations:

Proposition 3.3 *Let A be a recursive set.*

(i) A set C is a polynomial complexity core for A if and only if for every polynomial t and constant c, $\mathrm{ic}^t(x : A) > c$ *for almost all x in C.*

(ii) The set A is p-immune (resp. bi-immune) if and only if for every polynomial t and constant c, $\mathrm{ic}^t(x : A) > c$ *for almost all x in A (resp. Σ^*).*

Proof. Let us prove part (i); part (ii) then follows as a corollary. Assume first that for some polynomial t and constant c there are infinitely many x in C such that $\mathrm{ic}^t(x : A) \le c$. Then among the finitely many t-programs for A of size at most c there must be at least one, say q, for which $U(q, x) \neq \perp$ for infinitely many x in C. Let p_A be some fixed total program for A. Then $p = q \cup p_A$ is a total program for A, and for infinitely many x in C, $\mathrm{time}_p(x) \le \mathrm{time}_q(x) + \gamma|p| = O(t(|x|))$, showing that C cannot be a polynomial core for A.

Conversely, assume that C is not a polynomial core for A. Then there exist a total program p for A and a polynomial t such that for infinitely many x in C, $\mathrm{time}_p(x) \le t(|x|)$.

By adding a step counter, one can easily construct from p an interpreter M such that for some polynomial t' and all x, $\mathrm{time}_M(\lambda, x) \leq t'(|x|)$, $M(\lambda, x) \simeq A(x)$, and if $\mathrm{time}_p(x) \leq t(|x|)$, then $M(\lambda, x) \neq \perp$. Then for infinitely many x in C, $\mathrm{ic}_M^{t'}(x : A) = 0$, and by invariance there exist a polynomial t'' and a constant c such that for these x, $\mathrm{ic}^{t''}(x : A) \leq c$. □

Although the ic measure appears to be uncomputable (this is actually an open question), for recursive sets it can be approximated arbitrarily well. For any function r on the natural numbers, let us denote $r^{-1}(n) = \min\{k : r(k) \geq n\}$, and define an *$r$-bounded* instance complexity measure as follows:

$$r\text{-ic}^t(x : A) = \min\{|p| : U(p, x) \neq \perp, \text{ and } \text{for all } y, |y| \leq r(r^{-1}(|x|)), \mathrm{time}_U(p, y) \leq t(|y|) \text{ and } U(p, y) \simeq A(y).\}$$

Clearly $r\text{-ic}^t(x : A) \leq \mathrm{ic}^t(x : A)$ for all r, A, t, and x. Also, if there is a total T-program for A, and all of $r(r^{-1}(n))$, $t(n)$, and $T(n)$ are time-constructible and nondecreasing, then $r\text{-ic}^t(x : A)$ can be computed in time $O(n2^{r(r^{-1}(n))}T(n))$. (Note that if $t(n) = \omega(T(n))$, then $\mathrm{ic}^t(x : A)$, and hence also $r\text{-ic}^t(x : A)$, is bounded by a constant.)

Proposition 3.4 *For any nondecreasing time constructible function $r(n) \geq n$, there is a constant c such that for all A, nondecreasing $t(n) \geq n$, and x,*

$$\mathrm{ic}^{t'}(x : A) \leq r\text{-ic}^t(x : A) + r^{-1}(|x|) + c,$$

where $t'(n) = ct(n) \log t(n) + c$.

Proof. Let r be as stated, and consider an interpreter M that computes the following[3]:

$$M(\langle n, p\rangle, y) = \begin{cases} U(p, y), & \text{if } |y| \leq r(n), \\ \lambda, & \text{otherwise.} \end{cases}$$

Such an interpreter can easily be constructed so that for some small constant d, $\mathrm{time}_M(\langle n, p\rangle, y) \leq d(|y| + \mathrm{time}_U(p, y))$. Given then A, t, and x, let p_x be a minimal length r-ic^t-program for A deciding x, and let $p = \langle r^{-1}(|x|), p_x\rangle$. Then $M(p, x) = U(p_x, x) \neq \perp$, and for all y, $M(p, y) \simeq A(y)$ and

$$\mathrm{time}_M(p, y) \leq d(|y| + t(|y|)) \leq 2dt(|y|)$$

Thus,

$$\begin{aligned} \mathrm{ic}_M^{2dt}(x : A) &\leq |p| = |\langle r^{-1}(|x|), p_x\rangle| \\ &\leq |r^{-1}(|x|)| + |p_x| + 2\log|r^{-1}(|x|)| + 4 \\ &\leq r\text{-ic}^t(x : A) + r^{-1}(|x|) + 7. \end{aligned}$$

The result follows by invariance. □

We conclude this section with a simple, but very useful proposition on the behaviour of the ic measure under polynomial time reductions.

[3] Here, and also later in this paper, we occasionally equate natural numbers with their binary representations without leading zeros. Note that in this representation, $|n| \leq \log n + 1$.

Proposition 3.5 *Let f be a $\leq^p_{1-tt}$-reduction from a set A to a set B (more precisely, let f be the function mapping a string x to the one string queried in the reduction for x). Then there exists a constant c such that for any polynomial t there is a polynomial t' such that for all x,*

$$\mathrm{ic}^{t'}(x:A) \leq \mathrm{ic}^{t}(f(x):B) + c.$$

Proof. A $\leq^p_{1-tt}$-reduction from a set A to a set B consists of two polynomial time mappings $f : \Sigma^* \to \Sigma^*$ and $b : \Sigma^* \to \Sigma$, such that for all x, $x \in A$ if and only if $B(f(x)) = b(x)$. Assume that in the case under consideration, both of these mappings can be computed in time bounded by a nondecreasing polynomial $r(n)$. Let M be an interpreter implementing the following algorithm:

```
$M(q, x)$:

compute $y = f(x), b = b(x)$;
compute $z = U(q, y)$;
if $z = \perp$, then output $\lambda$,
else if $z = b$, then output 1,
else output 0.
```

Let t be any polynomial and x any string; w.l.o.g. assume that t is nondecreasing. It can be seen that if q is a t-program for B deciding $f(x)$, then q is also an (M, t'')-program for A deciding x, where $t''(n) = r(n) + t(r(n))$. Hence $\mathrm{ic}^{t''}_M(x:A) \leq \mathrm{ic}^t(f(x):B)$ for all x. But by invariance, there is a constant c, independent of t and t'', such that for all x, $\mathrm{ic}^{t'}(x:A) \leq \mathrm{ic}^{t''}(x:A) + c$, where $t'(n) = ct''(n)\log t''(n) + c$. □

It is quite straightforward to extend the above proof to yield also:

Proposition 3.6 *Let f be a $\leq^p_{btt}$-reduction from a set A to a set B (precisely, let f be the function mapping a string x to the set of strings queried in the reduction for x). Then there exists a constant c such that for any polynomial t there is a polynomial t' such that for all x,*

$$\mathrm{ic}^{t'}(x:A) \leq c \max_{y \in f(x)} \mathrm{ic}^t(f(x):B).$$

□

4 Sets with Logarithmic Instance Complexity

Recall that the class P can be characterized as the class of sets with constant-bounded instance complexity (w.r.t. polynomial time bounds); on the other hand, the instance complexity of any set can grow at most linearly. In this section, we study the class of sets with logarithmically bounded instance complexity. Our main result is that if a polynomially self-reducible set [23, 28] has logarithmic instance complexity then it is in P. Consequently SAT, and by application of Propositions 3.5 and 3.6, any set that is $\leq^p_{btt}$-hard for the class NP, can have logarithmic instance complexity only if P = NP. We also show that our class of sets lies properly between the advice complexity classes P/log and P/poly introduced by Karp and Lipton in [14], and is incomparable with the class P/lin.

Let us define, for functions $s(n), t(n)$,

$$\begin{aligned}\mathrm{IC}[s(n), t(n)] &= \{A : \mathrm{ic}^t(x : A) \leq s(n) \text{ for all } x\},\\ \mathrm{IC}[\log, \mathrm{poly}] &= \bigcup\{\mathrm{IC}[c \log n + c, t(n)] : \text{ constant } c, \text{ polynomial } t(n)\}.\end{aligned}$$

A set $A \subseteq \Sigma^*$ is *polynomially self-reducible* [23, 28] if there exist a well-founded partial order[4] $\preceq$ on Σ^*, and a polynomial time deterministic oracle Turing machine M, such that M with oracle A recognizes A, and M on any input x queries only strings that strictly precede x in the order $\preceq$. (For definitions of oracle machines and related notions see, e.g., [2].) Moreover, we require that there is a polynomial r such that if $x_0 \succ x_1 \succ \cdots \succ x_k$ is a descending chain in the query ordering, then $k \leq r(|x_0|)$, and $|x_i| \leq r(|x_0|)$ for every $i = 1, \ldots, k$.

Theorem 4.1 *Let A be a polynomially self-reducible set. Then $A \in$ IC[log, poly] if and only if $A \in$ P.*

Proof. The "if" direction follows from Proposition 3.2. For the "only if" direction, let M be the self-reducing machine for A, and let r be the associated chain-bounding polynomial. Assume that there are a constant c and a polynomial t such that for all $x \in \Sigma^*$, $\mathrm{ic}^t(x : A) \leq c \log |x| + c$. We claim that the recursive tree-pruning procedure given in Figure 1 is then a polynomial time algorithm for deciding membership in A.

To verify the correctness of the algorithm, consider a computation of it on an input x. Note that, by the assumption $A \in \mathrm{IC}[c \log n + c, t(n)]$, and the polynomial boundedness of the query chains, the global variable Π initially contains a set of polynomially many programs p such that for any string y queried by M during the computation, and for the original input x, there is some $p \in \Pi$ such that $\mathrm{time}_p(y) \leq t(|y|)$ and $U(p, y) = A(y)$. By induction on the recursion depth of the computation, one can then show that:

(i) whenever either one of the procedures *decide*(y) and *reduce*(y) returns a decision on whether a string y belongs to A, that decision is correct;

(ii) no t-programs for A are ever deleted from Π, so it is actually true throughout the computation that the programs in Π cover all the relevant strings, in the sense described above;

(iii) any call to either one of the procedures terminates.

The correctness of the algorithm follows from (i) and (iii); (ii) is an auxiliary observation needed for the induction.

To see that the computation actually terminates in polynomial time, note that whenever a call to *decide*(x) results in both the sets Π_0 and Π_1 becoming nonempty, the algorithm proceeds down a query chain, until at some level no further recursion is needed. Backing up from this point, the algorithm is able to eliminate at least one incorrect program from Π. Hence within a polynomial time of any moment that the routine *decide*(x) obtains an ambiguous answer from Π, at least one offending program from Π will be deleted. Since Π contains only polynomially many programs to start with, eventually *decide*(x) will obtain only unambiguous answers, and the procedure will terminate in polynomial time. $\square$

[4]A partial order $\preceq$ is *well-founded* if there are no infinite descending chains $x_0 \succ x_1 \succ x_2 \succ \cdots$, where $x_i \succ x_j$ means $x_j \preceq x_i$ and $x_j \neq x_i$.

```
On input $x$:

set $\Pi := \{p : |p| \leq c \log r(|x|) + c\}$;
return *decide*$(x)$.

*decide*$(x)$:

for every $p \in \Pi$, try to compute $U(p, x)$ in $t(|x|)$ steps;
    if this fails, set $\Pi := \Pi - \{p\}$;
let $\Pi_0$ and $\Pi_1$ be two variables local
    to this instantiation of *decide*$(x)$;
set $\Pi_0 := \{p \in \Pi : U(p, x) = 0\}$,
    $\Pi_1 := \{p \in \Pi : U(p, x) = 1\}$;
if $\Pi_0 = \emptyset$ then return 1,
else if $\Pi_1 = \emptyset$ then return 0,
else compute $a :=$ *reduce*$(x)$;
    if $a = 1$ then
        set $\Pi := \Pi - \Pi_0$,
        return 1;
    else
        set $\Pi := \Pi - \Pi_1$,
        return 0.

*reduce*$(x)$:

simulate $M$ on input $x$;
    whenever $M$ queries a string $y$,
    compute an answer to the query
    by recursively calling *decide*$(y)$;
if $M$ accepts $x$, then return 1,
else return 0.
```

Figure 1: A tree-pruning algorithm for a self-reducible set.

Corollary 4.2 *Assume* P $\neq$ NP*, and let* A *be a set that is* $\leq^p_{1-tt}$*-hard for* NP*. Then* $A \notin$ IC[log, poly].

Proof. By Theorem 4.1, SAT $\in$ IC[log, poly] implies that P = NP. Let us assume that some $\leq^p_{1-tt}$-hard set A is in IC[log, poly]; we show that this implies that also SAT $\in$ IC[log, poly]. Let c be a constant and t a polynomial such that for all x, $\mathrm{ic}^t(x : A) \leq c \log |x| + c$. Let f be a $\leq^p_{1-tt}$-reduction from SAT to A, and let d be a constant such that $|f(\phi)| \leq |\phi|^d$ for all ϕ. Then, by Proposition 3.5, there exist a polynomial t' and a constant c' such that for all ϕ,

$$\begin{aligned} \mathrm{ic}^{t'}(\phi : \mathrm{SAT}) &\leq \mathrm{ic}^t(f(\phi) : A) + c' \\ &\leq c \log |f(\phi)| + c + c' \\ &\leq cd \log |\phi| + c + c' \\ &\leq c'' \log |\phi| + c'', \end{aligned}$$

where $c'' = \max\{cd, c + c'\}$. □

Using Proposition 3.6, the result can be extended also to sets that are NP-hard with respect to $\leq^p_{btt}$-reductions.

Interestingly, we can show that the class IC[log, poly] is located properly between the advice complexity classes P/log and P/poly introduced by Karp and Lipton in [14]. Thus, Theorem 4.1 yields a provable strengthening of Karp's and Lipton's result that SAT $\in$ P/log if and only if P = NP.

Definition 4.1 (Karp, Lipton) Let f be a function on the natural numbers. A set A belongs to the class P$/f$ if there exist another set $B \in$ P and a function $h : \mathcal{N} \to \Sigma^*$, such that for all n, $|h(n)| \leq f(n)$, and for all x, $x \in A$ if and only if $\langle x, h(|x|)\rangle \in B$. We define

$$\begin{aligned} \mathrm{P/log} &= \bigcup_{c>0} \mathrm{P}/c \log n, \\ \mathrm{P/lin} &= \bigcup_{c>0} \mathrm{P}/cn, \\ \mathrm{P/poly} &= \bigcup_{c>0} \mathrm{P}/n^c. \end{aligned}$$

Theorem 4.3 *(i)* P/log $\subseteq$ IC[log, poly] $\subseteq$ P/poly*;*

(ii) P/lin $\not\subseteq$ IC[log, poly]*;*

(iii) IC[log, poly] $\not\subseteq$ P$/n^c$ *for any fixed* $c > 0$.

Proof. (i) Given $A \in$ P/log, let $B \in$ P and $h : \mathcal{N} \to \Sigma^*$, $|h(n)| \leq c \log n$, be such that for all x, $A(x) = B(\langle x, h(|x|)\rangle)$. Let M be an interpreter that on input $(\langle n, z\rangle, x)$ outputs the value $B(\langle x, z\rangle)$ if $|x| = n$, and λ otherwise. Clearly, for some polynomial t and all x, $|x| = n$, $\mathrm{ic}^t_M(x : A) \leq |n| + c \log n + 2 \log |n| + 4 = O(\log n)$. By invariance, then, there exist a polynomial t' and a constant c' such that for all x, $\mathrm{ic}^{t'}(x : A) \leq c' \log |x| + c'$. Hence $A \in$ IC[log, poly].

Let then A be a set in IC[log, poly], and let c be a constant and t a polynomial such that for all x, $\mathrm{ic}^t(x : A) \leq c \log |x| + c$; w.l.o.g. assume that $t(n) \geq n^c \log n$. For any given

n, let $p_1, \ldots, p_k$ be a listing of all the t-programs for A of size at most $c \log n + c$. Then $p = p_1 \cup \ldots \cup p_k$ is a program for A such that $U(p,x) \neq \perp$ for all x, $|x| \leq n$. For the size of p, we obtain the bound

$$\begin{aligned} |p| &\leq \sum_{i=1}^{k} |p_i| + 2\sum_{i=1}^{k} \log |p_i| + k\gamma \\ &\leq 2^c n^c (c \log n + c + 2\log(c\log n + c) + \gamma) \\ &= O(n^c \log n), \end{aligned}$$

and for its time complexity the bound

$$\begin{aligned} \mathrm{time}_p(x) &\leq \sum_{i=1}^{k} \mathrm{time}_{p_i}(x) + k\gamma|p| \\ &\leq 2^c n^c (t(n) + \gamma|p|) \\ &= O(n^c t(n)). \end{aligned}$$

For a given n, let $p^{(n)}$ denote the program defined above, and let r be a polynomial bounding the running times of all such $p^{(n)}$. Define $h(n) = p^{(n)}$, and let $B(\langle x,p\rangle) = 1$ if $U(p,x) = 1$ in $r(|x|)$ steps, and 0 otherwise. Then clearly h and B satisfy the conditions of Definition 4.1 for showing that $A \in \mathrm{P/poly}$.

(ii) To see that P/lin $\not\subseteq$ IC[log, poly], let A be a set that for each n contains exactly one string x of length n, and this x is such that $K(x) \geq n$. (Here $K(x)$ denotes the standard time-unbounded Kolmogorov complexity of x.) Clearly $A \in$ P/lin; to show that $A \notin$ IC[log, poly], assume to the contrary that there are a constant c and a polynomial t such that for all x, $U(p_x, x) = A(x)$ in time $t(|x|)$ for some program p_x for A, $|p_x| \leq c \log |x| + c$. Let M be an interpreter implementing the following algorithm:

```
$M(\langle n,p\rangle, \lambda)$:

for all $x$, $|x| = n$, do:
    simulate $U(p,x)$ for $t(n)$ steps;
    if in this time $U(p,x) = 1$,
    then output $x$ and halt.
```

Now if $x \in A$, $|x| = n$, then $M(\langle n,p_x\rangle, \lambda) = x$, and so $K_M(x) \leq |\langle n,p_x\rangle| = O(\log n)$. Hence, by invariance, also $K(x) = O(\log n)$. But by the construction of A, for large enough x this is not possible. (In fact, doing the argument in a little more detail shows that for every polynomial t, $\mathrm{ic}^t(x : A) > |x| - 2\log|x|$ for almost all $x \in A$. Also, the class P/lin can, with minor modifications, be replaced by any class P$/f$, where $f(n) = \omega(\log n)$.)

(iii) Let some fixed $c > 0$ be given; for simplicity, assume that c is an integer. We show how to construct by diagonalization a set A such that $A \in$ IC[log, poly], but $A \notin$ P$/f$ for any $f(n) < n^c$. Let $B_1, B_2, \ldots$ be some enumeration of all sets in P in which every set appears infinitely often. At stage n of the construction, we diagonalize against basis set B_n and all advice strings w, $|w| < n^c$, as follows. Let Σ^n denote the set of strings of length n; w.l.o.g. assume that $2^n \geq n^c$. Let $x_1, x_2, \ldots, x_{2^n}$ be an enumeration of the strings in Σ^n in lexicographic order, and let S_n denote the set $\{x_1, x_2, \ldots, x_{n^c}\}$. For each string w, $|w| < n^c$, let $A_w = \{x \in S_n : \langle x,w\rangle \in B_n\}$. Since S_n has 2^{n^c} different subsets, but there are fewer than this number of sets A_w, there is some $A^{(n)} \subseteq S_n$ such that $A^{(n)} \neq A_w$ for all w, $|w| < n^c$. Define A as the union of the $A^{(n)}$ sets from each stage, $A = \bigcup_{n \geq 0} A^{(n)}$.

By construction, $A \notin \mathrm{P}/n^c$; let us show that $A \in \mathrm{IC}[\log, \mathrm{poly}]$. Consider an interpreter M implementing the following algorithm:

```
$M(\langle n, \langle k, d \rangle \rangle, x)$:

if $|x| \neq n$ then output $\lambda$ and halt;
let $x = x_i$ in the enumeration of $\Sigma^n$;
if $i > n^c$ then output 0,
else if $i = k$ then output $d$, else output $\lambda$.
```

Given an x such that $|x| = n$, and $x = x_k$ in the enumeration of Σ^n, define

$$p_x = \begin{cases} \langle n, \langle k, A(k) \rangle \rangle, & \text{if } k \leq n^c, \\ \langle n, \langle 0, 0 \rangle \rangle, & \text{if } k > n^c. \end{cases}$$

Clearly there is some (low-order) polynomial t such that for every x, p_x is an (M, t)-program for A deciding x. Moreover, for x such that $|x| = n$,

$$\begin{aligned} |p_x| &\leq |n| + |n^c| + 1 + 2\log|n| + 2\log|n^c| + 8 \\ &= O(\log n). \end{aligned}$$

The result follows by invariance. □

Corollary 4.4 *(i)* $\mathrm{P}/\log \subsetneq \mathrm{IC}[\log, \mathrm{poly}] \subsetneq \mathrm{P}/\mathrm{poly}$*;*

(ii) $\mathrm{P}/\mathrm{lin} \not\subseteq \mathrm{IC}[\log, \mathrm{poly}]$ *and* $\mathrm{IC}[\log, \mathrm{poly}] \not\subseteq \mathrm{P}/\mathrm{lin}$. □

5 Hard Instances

In this section, we prove three theorems in partial support of the "instance complexity conjecture" outlined in the Introduction, claiming that any set A not recognizable within a given time bound t will have infinitely many strings whose instance complexity with respect to A and t will be close to their Kolmogorov complexity. Our first theorem applies to an arbitrary recursive set A, but leaves a rather large gap between the time bounds for the respective instance and Kolmogorov complexities — moreover, the size of the gap depends on the complexity of recognizing A. The second theorem, using a more involved construction, narrows the time bound gap to a polynomial in the interesting special cases of NP-hard and DEXT-hard sets. The third result proves the existence of sets with respect to which *all* strings have instance complexity close to their Kolmogorov complexity.

Theorem 5.1 *Let $s(n) \geq t(n) \geq n$ be nondecreasing time constructible functions, and let A be a set in $\mathrm{DTIME}(s) - \mathrm{DTIME}(t)$. Then there exists a constant c such that for infinitely many x,*

$$\mathrm{ic}^t(x : A) \geq K^{s'}(x) - c,$$

where $s'(n) = c2^{2n}s(n)(n + \log s(n))$.

Proof. The basic idea of the proof is to use the ic^t-complexity of a string to upper bound its K-complexity. A straightforward approach to doing this would be to take a t-program p for A, and convert it to a program p' that produces some string x for which p is minimal, i.e., such that no smaller t-program for A decides x. This direct scheme, however, fails due to the fact that it is undecidable whether a given program is a t-program for A; instead, we have to resort to an indirect argument via *bounded* ic^t-programs.

For any string x, let $\hat{p}_x$ denote the minimal *lin*-ic^t-program for A deciding x, i.e.

$$\begin{aligned}\hat{p}_x = {} & \text{lexicographically first } p \text{ such that } U(p,x) \neq \bot, \text{ and} \\ & \text{for all } y,\ |y| \leq |x|,\ \mathrm{time}_U(p,y) \leq t(|y|) \text{ and } U(p,y) \simeq A(y).\end{aligned}$$

Correspondingly, let p_x denote the true minimal t-program for A deciding x; then $|\hat{p}_x| \leq |p_x| = \mathrm{ic}^t(x : A)$. Let M be an interpreter that computes:

$$M(p,\lambda) = \text{lexicographically first } z,\ |z| \geq |p|, \text{ such that } \hat{p}_z = p \text{ (if such a } z \text{ exists).}$$

It is fairly straightforward to construct M so that $\mathrm{time}_M(p,\lambda) \leq 2^{2n}s(n)$, where n is the length of the output $z = M(p,\lambda)$. Hence if M produces output z from program p, we have

$$K_M^{2^{2n}s(n)}(z) \leq |p|.$$

By invariance, there is a constant c such that for all z,

$$K^{s'(n)}(z) \leq K_M^{2^{2n}s(n)}(z) + c,$$

where $s'(n) = c2^{2n}s(n)(n + \log s(n))$.

Assume then that the statement of the theorem does not hold, so that there is some x_0 such that for all $x > x_0$ (in the lexicographic ordering),

$$\mathrm{ic}^t(x : A) < K^{s'}(x) - c.$$

Denote

$$\Pi = \{\hat{p}_x : x \leq x_0\}.$$

We claim that for some $x > x_0$, $\hat{p}_x \notin \Pi$. This is because otherwise the programs in Π could decide all strings correctly in time t, contrary to the assumption that $A \notin \mathrm{DTIME}(t)$. More precisely, assume that the claim does not hold, and denote

$$\begin{aligned}\Pi^\infty &= \{\hat{p} \in \Pi : \hat{p} = \hat{p}_x \text{ for infinitely many } x\}, \\ X &= \{x : \hat{p}_x \notin \Pi^\infty\}.\end{aligned}$$

Then every program $\hat{p}_x \in \Pi^\infty$ is in fact a t-program for A (because it is a *lin*-ic^t-program deciding arbitrarily long strings x), and the set X is finite. Let $\hat{p}_1, \ldots, \hat{p}_k$ be the programs in Π^∞, and let p_X be an $O(n)$-program for X. Now the program

$$\hat{p}_1 \cup \ldots \cup \hat{p}_k \cup p_X$$

is an $O(t)$-program for A.

But let $x > x_0$ be such that $\hat{p}_x \notin \Pi$, and denote $z = M(\hat{p}_x, \lambda)$. Then $\hat{p}_z = \hat{p}_x$, and because $\hat{p}_z \notin \Pi$, also $z > x_0$. Hence

$$K^{s'}(z) \leq |\hat{p}_z| + c \leq |p_z| + c = \mathrm{ic}^t(z : A) + c < K^{s'}(z).$$

From the contradiction it follows that for the chosen value of c,

$$\mathrm{ic}^t(x : A) \geq K^{s'}(x) - c$$

holds for infinitely many x. □

Corollary 5.2 *Let A be a set in* DEXT − P*. Then there exists a constant c such that for any polynomial t there exist a constant c_t and infinitely many x, such that*

$$\mathrm{ic}^t(x : A) \geq K^{2^{cn}}(x) - c_t.$$

□

Before presenting the next theorem, on intrinsically hard instances for NP-hard and DEXT-hard sets, we need to introduce a new "structural complexity" property.

Definition 5.1 Let t be a time bound. A set S is *t-coverable within* a set A if there is a set $E \in \mathrm{DTIME}(t)$ such that $A \cap S \subseteq E \subseteq A$. A set S is *almost t-coverable within A* if there is a set $E \subseteq A, E \in \mathrm{DTIME}(t)$, such that for any other $E' \subseteq A, E' \in \mathrm{DTIME}(t)$, the set $(E' - E) \cap S$ is finite.

The notion of almost t-coverability is a generalization of the notion of almost t-immunity discussed (for polynomial t) in [24], and under the name "non-t-levelability" in [25]. A set A is almost t-immune if it contains a DTIME(t) subset E that is maximal in the sense that for any other $E' \subseteq A$, $E' \in \mathrm{DTIME}(t)$, the set $E' - E$ is finite. Hence A is almost t-immune if and only if it is almost t-coverable within itself.

A set A is *paddable* if there is a polynomial time computable one-to-one function $\mathrm{pad}(x, y)$ such that for any strings x, y, $\mathrm{pad}(x, y) \in A$ if and only if $x \in A$. A is *honestly* paddable if for some constant k, $|\mathrm{pad}(x, y)| \geq (|x| + |y|)^{1/k}$ for all x, y. A is *linearly* paddable if for some constant k, $k^{-1}(|x| + |y|) \leq |\mathrm{pad}(x, y)| \leq k(|x| + |y|)$ for all x, y. We note that many natural NP- and DEXT-complete sets are linearly paddable (e.g., the NP-complete set SAT, and the DEXT-complete set of circular attribute grammars [13]).

The main rationale for Definition 5.1 lies in the following result, essentially due to Hartmanis [10]. For functions $s(n), t(n)$, define

$$K[s(n), t(n)] = \{x : K^t(x) \leq s(|x|)\}.$$

Lemma 5.3 (Hartmanis) *If* DEXT $\neq$ NEXT, *then $K[c \log n, n^c]$ is not t-coverable within* SAT, *for any constant $c \geq 2$ and polynomial t.*

Proof. Using the honest (in fact, linear) paddability of SAT, it is easy to show that for any $c \geq 2$, the set SAT $\cap\, K[c \log n, n^c]$ is $\leq^p_m$-hard for the class of tally sets in NP. If there is a set $E \in \mathrm{DTIME}(t) \subseteq \mathrm{P}$ such that SAT $\cap\, K[c \log n, n^c] \subseteq E \subseteq$ SAT, then in fact SAT $\cap\, K[c \log n, n^c] \in$ P, and so there cannot be any tally sets in NP − P; hence DEXT = NEXT [5]. □

One can also easily show that if A is honestly paddable and $\leq^p_m$-hard for DEXT, then $A \cap K[c \log n, n^c], c \geq 2$, is $\leq^p_m$-hard for the class of tally sets in DEXT. Since tally sets provably exist in DEXT − P, this establishes without any assumptions that $K[c \log n, n^c]$ cannot be t-covered within A for any polynomial t.

By our next lemma, we can improve the above results from "not t-coverable" to "not almost t-coverable" for any linearly paddable set A.

Lemma 5.4 *Let A be a linearly paddable set. Then for all sufficiently large constants c and polynomials t, $K[c\log n, n^c]$ is almost t-coverable within A if and only if it is t-coverable within A.*

Proof. The "if" direction is trivial. To prove the "only if" direction, we apply a construction from [25]. Let A be a linearly paddable set, with a padding function $\mathrm{pad}(x, y)$ that is computable in time $O((|x| + |y|)^l)$, and is such that $k^{-1}(|x| + |y|) \leq |\mathrm{pad}(x, y)| \leq k(|x| + |y|)$. Consider a function $f(x)$ defined as $f(x) = \mathrm{pad}(x, 1^{2k|x|})$. Clearly $f(x)$ can be computed in time $O(|x|^l)$, and has the property that $|f(x)| \geq 2|x|$. For definiteness, let us assume w.l.o.g. that $f = f_p$ for some program p such that $\mathrm{time}_p(x) \leq |x|^l$ for all x.

Assume, for a contradiction, that for arbitrarily large c, d, $K[c\log n, n^c]$ is almost n^d-coverable within A, but not n^d-coverable within A. Choose some c, d with this property large enough so that

$$\begin{aligned}|p| + c\log\frac{n}{2} + 2\log|p| + \gamma &\leq c\log n,\\ (\frac{n}{2})^c + (\frac{n}{2})^l + \gamma c\log n &\leq n^c,\end{aligned}$$

and $d > l$. Let E be a maximal partial $\mathrm{DTIME}(n^d)$-cover (as per Definition 5.1) for $K = K[c\log n, n^c]$ within A. Since K is not n^d-coverable within A, the set $(A \cap K) - E$ is infinite.

Consider a string $x \in K$, $x \neq \lambda$, and let q be a program of size at most $c\log|x|$ that computes x from λ in time $|x|^c$. Then the image y of x under $f = f_p$ can be computed from λ by the program $p \circ q$, for which

$$\begin{aligned}|p \circ q| &\leq |p| + c\log|x| + 2\log|p| + \gamma,\\ \mathrm{time}_{p\circ q}(\lambda) &\leq |x|^c + |x|^l + \gamma|p \circ q|.\end{aligned}$$

Since $|y| \geq 2|x|$, it follows from our assumptions on c that also $y \in K[c\log n, n^c] = K$. Hence for any $x \in A \cap K$, $x \neq \lambda$, the set

$$E_x = \{x, f(x), f(f(x)), \ldots\}$$

is an infinite subset of $A \cap K$. Moreover, there is a program that decides whether a string y is in E_x in time $O(|y|^l \log|y|) = O(|y|^d)$, so $E_x \in \mathrm{DTIME}(n^d)$. By the maximality of E, then, $E_x - E$ is finite. In particular, for each of the infinitely many $x \in (A \cap K) - E$ there is a y, $|y| \geq |x|$, such that $y \in (A \cap K) - E$ and $f(y) \in E$. Hence, the set

$$B = \{y : y \notin E, f(y) \in E\}$$

contains infinitely many strings that are in $A \cap K$ but not in E. But B is a subset of A (because $y \in A$ if and only if $f(y) \in A$), and $B \in \mathrm{DTIME}(n^d)$ (by the closure of $\mathrm{DTIME}(n^d)$ under Boolean operations and the fact that $|f(x)| = O(|x|)$); this contradicts the assumed maximality of E. $\square$

Now we are in a position to state and prove our second main theorem. As the statement of the theorem is rather technical, the reader may wish to glance at the several corollaries following the result for motivation.

Theorem 5.5 *Let A be a recursive set, and let $s(n), u(n)$, and $t(n)$ be nondecreasing functions such that $2^{s(n)}, u(n)$, and $t(n)$ are time constructible. Assume that the set $K[s(n), u(n)]$ is not almost t-coverable within A. Then for any time constructible $t'(n) = \omega(n\, 2^{s(n)}\, (t(n) + u(n)))$, there exists a constant c such that for infinitely many x,*

$$\mathrm{ic}^t(x : A) \geq K^{t''}(x) - c,$$

where $t''(n) = ct'(n) \log t'(n) + c$.

Proof. Let p be any program for A, and τ a time bound. Let $\mu^\tau(p)$ denote the set of strings for which p is "τ-minimal" in the following sense:

$$\begin{aligned}\mu^\tau(p) = \{x : {}& U(p,x) \neq \bot, \mathrm{time}_p(x) \leq \tau(|x|),\\ & U(q,x) = \bot \text{ for all } q < p \text{ that are } \tau\text{-programs for } A.\}\end{aligned}$$

Note that if $x \in \mu^\tau(p)$, then $\mathrm{ic}^\tau(x : A) \geq |p|$.

In outline, the proof now proceeds as follows. Assume first that p is a dt-program for A for some constant $d \geq 1$, and that $\mu^{dt}(p)$ has infinite intersection with the set $K = K[s(n), u(n)]$. We show that in this case p can be turned, with a constant c increase in size, into a t''-program for producing some string $x \in \mu^{d't}(p) \cap K$, for some $1 \leq d' \leq d$. For such an x, it is then the case that

$$K^{t''}(x) \leq |p| + c \leq \mathrm{ic}^{d't}(x : A) + c \leq \mathrm{ic}^t(x : A) + c.$$

To conclude our result, we finally argue that if K is not almost t-coverable within A, there must exist infinitely many $O(t)$-programs p for A such that the associated $\mu^{dt}(p) \cap K$ sets are infinite.

Let p_A be some fixed total program for A. We claim that the algorithm presented in Figure 2 produces, whenever it halts, from a given dt-program p for A a string $x \in \mu^{d't}(p) \cap K$, for some $1 \leq d' \leq d$. Let M be an interpreter implementing the algorithm. To verify that M behaves as desired, observe first that if $M(p, \lambda)$ halts and prints out some string x, then it does so in at most $|x| 2^{s(|x|)+1} t_0(|x|) = t'(|x|)$ steps. Hence for any such x, $K_M^{t'}(x) \leq |p|$. Moreover, if p is a dt-program for A deciding x, then the check performed in the innermost loop of the algorithm ensures that for some d', $1 \leq d' \leq d$, no $q < p$ can be a $d't$-program for x; thus $x \in \mu^{d't}(p)$ and $\mathrm{ic}^{d't}(x : A) \geq |p|$. Hence, by invariance there is a constant c, independent of x, such that

$$\begin{aligned} K^{t''}(x) &\leq K_M^{t'}(x) + c &&\leq |p| + c \\ &\leq \mathrm{ic}^{d't}(x : A) + c &&\leq \mathrm{ic}^t(x : A) + c, \end{aligned}$$

where $t''(n) = ct'(n) \log t'(n) + c$.

Let us then show that the computation $M(p, \lambda)$ indeed does halt for any p such that p is a dt-program for A and $\mu^{dt}(p) \cap K$ is infinite. Note that since $t_1(n) = \omega(t(n))$, the function $\sqrt{t_1(n)/t(n)}$ tends to infinity, and so for large enough n, the appropriate value of d is always tried out in the second-innermost loop of the algorithm. For each of the finitely many $q < p$ that are not dt-programs for A, there is some string z_q such that either $\mathrm{time}_U(q, z_q) > dt(|z_q|)$ or $U(q, z_q) \not\simeq U(p_A, z_q)$. Let θ be a nondecreasing function such that $\mathrm{time}_{p_A}(z) \leq \theta(|z|)$ for all z, and let

$$n_0 = \max\{|z_q| : q < p \text{ is not a } dt\text{-program for } A\}.$$

```
$M(p, \lambda)$:

for $n = 0, 1, 2, \ldots$ do:
    for all $y, |y| \leq s(n)$, do:
        for $t_0(n) = t'(n)/n2^{s(n)+1}$ steps, try to do the following:

            $\{$find a string $x \in K$ of length $n\}$
            in $u(n)$ steps, try to compute $x = U(y, \lambda)$;
            if time runs out or $|x| \neq n$, go to next $y$;

            $\{$verify that $x \in \mu^{dt}(p)$, for some $d\}$
            let $t_1(n) = t_0(n) - u(n)$;
            for $d = 1, 2, \ldots, \sqrt{t_1(n)/t(n)}$, do:
                for $\sqrt{t_1(n)t(n)}$ steps, do:

                    $\{$verify that $U(p, x) \neq \perp$ in time $dt(n)\}$
                    in $dt(n)$ steps, try to compute $U(p, x)$;
                    if time runs out, go to next $d$;
                    if $U(p, x) = \perp$, go to next $y$;

                    $\{$verify that $p$ is $dt$-minimal for $x\}$
                    in the remaining time, try to do the following:
                    for all $q < p$ do:
                        if $\text{time}_U(q, x) \leq dt(|x|)$ and $U(q, x) \neq \perp$
                        then check, in lexicographic order, that
                        for some string $z$ either $\text{time}_U(q, z) > dt(|z|)$ or
                        $U(q, z) \not\simeq U(p_A, z)$.

                    if the last check can be successfully completed,
                    then output $x$ and halt.
```

Figure 2: An algorithm for producing a “hard instance” for a set.

Then the time required to complete the minimality check in the innermost loop is, for $|x| = n$,

$$O(2^{|p|}(dt(n) + 2^{n_0+1}(dt(n_0) + \theta(n_0)))) = O(t(n)).$$

But the time available for the check is $\sqrt{t_1(n)t(n)} - dt(n) = \omega(t(n))$, so for some sufficiently large $x \in \mu^{dt}(p) \cap K$ the test will be successfully completed, and x printed — unless some $x' \in \mu^{d't}(p) \cap K$, $d' \leq d$, gets printed first.

It remains to be shown that if $K = K[s(n), u(n)]$ cannot be almost t-covered within A, then there will be infinitely many programs p of the desired type. Assume to the contrary that there is some p_0 such that for any constant $d \geq 1$ and any $p > p_0$ that is a dt-program for A, the set $\mu^{dt}(p) \cap K$ is finite. Let $q_1, \ldots, q_k$ be all the $O(t)$-programs for A up to, and possibly including, p_0. Define $q_0 = q_1 \cup \ldots \cup q_k$. We claim that then $L_{q_0} = f_{q_0}^{-1}(1)$ almost t-covers K within A.

Clearly $f_{q_0}(x) \simeq A(x)$ for all x, and by the efficient closure under union of our programming system, $\text{time}_{q_0}(x) = O(t(|x|))$. Hence $L_{q_0} \subseteq A$, and $L_{q_0} \in \text{DTIME}(t)$. Assume, for a contradiction, that for some program r such that $L_r \subseteq A$ and $\text{time}_r(x) \leq dt(|x|)$ for some constant $d \geq 1$, there are infinitely many strings in $(L_r - L_{q_0}) \cap K$. W.l.o.g., assume that $f_r(x) = \perp$ for all $x \notin L_r$. Then r is a dt-program for A such that for infinitely many $x \in K$, $U(q_0, x) = \perp$ but $U(r, x) \neq \perp$. Each of these $x \in A \cap K$ belongs to $\mu^{dt}(r')$ for some dt-program r' for A, $p_0 < r' \leq r$. Hence there must exist some dt-program r' for A, $p_0 < r' \leq r$, such that $\mu^{dt}(r') \cap K$ is infinite. But by the definition of p_0, this is impossible. □

For brevity, let us say that a set A *has p-hard instances* if for any polynomial t there exist a polynomial t' and a constant c such that for infinitely many x, $\text{ic}^t(x : A) \geq K^{t'}(x) - c$. The theorem immediately implies the following corollaries:

Corollary 5.6 *If* DEXT $\neq$ NEXT, *then* SAT *has p-hard instances.*

Proof. By Lemma 5.3, Lemma 5.4, and Theorem 5.5. □

Corollary 5.7 *Any linearly paddable* DEXT*-complete set has p-hard instances.*

Proof. By the discussion following Lemma 5.3, Lemma 5.4, and Theorem 5.5. □

We can translate these results upwards using the following lemma:

Lemma 5.8 *If A has p-hard instances, and $A \leq^p_{1-tt} B$, then B has p-hard instances.*

Proof. Assume that A has p-hard instances, and let f be a $\leq^p_{1-tt}$-reduction from A to B (precisely, f is the function mapping a string x to the string queried in the reduction for x). Observe that because f is polynomial time computable, there is a constant e such that for any polynomial u there is a polynomial u' such that for all x,

$$K^{u'}(f(x)) \leq K^u(x) + e. \tag{1}$$

This follows from the efficient closure under composition of our programming system (or also just by invariance).

To show that B has p-hard instances, fix some polynomial t. Then, by Proposition 3.5, there exist a polynomial t' and a constant c such that for all x,

$$\text{ic}^{t'}(x : A) \leq \text{ic}^t(f(x) : B) + c.$$

The assumption that A has p-hard instances, on the other hand, implies that for some polynomial t'' and constant d, there exist infinitely many x such that

$$\mathrm{ic}^{t'}(x : A) \geq K^{t''}(x) - d.$$

Combining these, we see that for infinitely many x,

$$\mathrm{ic}^{t}(f(x) : B) \geq K^{t''}(x) - (c + d). \tag{2}$$

Applying now inequality (1), we obtain that for some polynomial t''' and constant e, and for infinitely many x,

$$\mathrm{ic}^{t}(f(x) : B) \geq K^{t'''}(f(x)) - (c + d + e).$$

Our result is complete, when we observe that inequality (2) implies that for the infinitely many x's we are considering, there must also be infinitely many different values of $f(x)$. □

Corollary 5.9 *If* DEXT $\neq$ NEXT, *then any set that is* $\leq^{p}_{1-tt}$*-hard for* NP *has p-hard instances.*

Proof. By Corollary 5.6 and Lemma 5.8. □

Corollary 5.10 *Any set that is* $\leq^{p}_{1-tt}$*-hard for* DEXT *has p-hard instances.*

Proof. By Corollary 5.7 and Lemma 5.8, and the fact that linearly paddable DEXT-complete sets exist. □

Our third main theorem, and its corollary concern the existence of dense sets of relatively hard instances for sets in DEXT.

Theorem 5.11 *There exists a set* $A \in$ DEXT *such that for some constant* c *and all* x,

$$\mathrm{ic}^{\exp}(x : A) \geq K^{\exp'}(x) - 2\log K^{\exp'}(x) - c,$$

where $\exp(n) = 2^n$ *and* $\exp'(n) = cn2^{2n} + c$.

Proof. The set A is constructed by a "weighted diagonalization" [22, 31] over all 2^n time bounded programs. The construction proceeds in stages corresponding to all strings $x \in \Sigma^*$, in lexicographic order. Initially $A = \emptyset$, and it is then decided at stage x whether $x \in A$.

Conceptually, each program p is initially assigned a *weight* of $w(p) = 2^{-(2|p|+1)}$. At each stage x in the construction, some set Π of the programs are "alive"; initially, the set Π contains all programs. In the course of the construction, the weights of some programs are increased, but at the same time programs are eliminated from Π so that at all stages, $\sum_{p\in\Pi} w(p) \leq 1$. (Note that this is true in the beginning.) The algorithm for stage x is given in Figure 3.

Clearly $A \in$ DEXT. (In fact, computing the construction up to stage x, $|x| = n$, can be done in time $O(2^{3n})$; by invariance, there is then a total $O(n2^{3n})$-program for A.) Note also that the upper bound on the total weight of programs in Π is maintained: at each stage, a total weight equal to $\min\{w_0, w_1\}$ is added, but before this, a set of programs with total weight equal to or greater than this has been eliminated.

Let $\Pi^{(x)}$ denote the set of programs in Π at the completion of stage x, and let $\hat{\Pi} = \bigcap_x \Pi^{(x)}$.

Stage x:

```
let $n = |x|$;
set $\Pi_0 := \{p \in \Pi : |p| \leq n, U(p,x) = 0 \text{ in } 2^n \text{ steps}\}$,
    $\Pi_1 := \{p \in \Pi : |p| \leq n, U(p,x) = 1 \text{ in } 2^n \text{ steps}\}$;
set $w_0 := \sum_{p \in \Pi_0} w(p)$,
    $w_1 := \sum_{p \in \Pi_1} w(p)$;
if $w_0 \geq w_1$, then
    set $A := A \cup \{x\}$;
    set $\Pi := \Pi - \Pi_0$;
    for every $p \in \Pi_1$, set $w(p) := 2w(p)$
else
    set $\Pi := \Pi - \Pi_1$;
    for every $p \in \Pi_0$, set $w(p) := 2w(p)$.
```

Figure 3: Stage construction for an everywhere-hard set.

We claim that

(i) if p is a program for A, then $p \in \hat{\Pi}$; and

(ii) if $p \in \hat{\Pi}$, then the set $E(p) = \{x : |x| \geq |p|, \mathrm{time}_p(x) \leq 2^{|x|}\}$ has at most $2|p|+1$ members.

To see (i), assume that $p \notin \hat{\Pi}$. Then p must have been eliminated from Π at some stage x. But by construction, then, $U(p,x) \not\simeq A(x)$. For (ii), note that for every x, $|x| \geq |p|$, such that $U(p,x) = A(x)$ in $2^{|x|}$ steps, the weight of p is doubled. Because the initial weight of $|p|$ is $2^{-(2|p|+1)}$, and the total weight of all programs is bounded by 1, this doubling can occur at most $2|p|+1$ times.

Consider then an interpreter M that on input $(\langle k,p\rangle, \lambda)$ outputs the lexicographically kth string x in $E(p)$, whenever $E(p)$ contains at least k strings, and does not halt otherwise. Such an M can easily be implemented so that when M halts with output x, then $\mathrm{time}_M(\langle k,p\rangle,\lambda) \leq 2^{2|x|}$. Since $k \leq 2|p|+1$ for every $p \in \hat{\Pi}$ and $x \in E(p)$, it follows that in this case

$$|k| \leq \log k + 1 \leq \log(2|p|+1) + 1 \leq \log|p| + 3,$$

and hence

$$\begin{aligned} K_M^{2^{2n}}(x) &\leq |\langle k,p\rangle| \leq |k| + |p| + 2\log|k| + 4 \\ &\leq |p| + \log|p| + 2\log\log|p| + 11. \end{aligned}$$

By invariance, then, there is a constant c such that for all $p \in \hat{\Pi}$, $x \in E(p)$,

$$K^T(x) \leq |p| + 2\log|p| + c', \tag{3}$$

where $T(n) = c'n2^{2n} + c'$. Let $c \geq c'$ be a constant such that for all strings x,

$$|x| \geq K^{\mathrm{exp}'}(x) - 2\log K^{\mathrm{exp}'}(x) - c, \tag{4}$$

where $\exp'(n) = cn2^{2n} + c$. Note that because $c \geq c'$ and $\exp'(n) \geq T(n)$, by (3) it is also true that for all $p \in \hat{\Pi}$, $x \in E(p)$,

$$K^{\exp'}(x) \leq |p| + 2\log|p| + c. \qquad (5)$$

Let then x be any string, and let p be a minimal length exp-program for A deciding x. To establish our result, we need to consider two cases.

(i) If $|x| < |p|$, then by (4),

$$\mathrm{ic}^{\exp}(x : A) = |p| > |x| \geq K^{\exp'}(x) - 2\log K^{\exp'}(x) - c.$$

(ii) If $|x| \geq |p|$, then $x \in E(p)$, and (5) easily implies that

$$\mathrm{ic}^{\exp}(x : A) = |p| \geq K^{\exp'}(x) - 2\log K^{\exp'}(x) - c. \ \Box$$

Let $\Sigma^{(n)}$ denote the set of strings of length at most n. A set of strings C is *exponentially dense* if there is a constant $\epsilon > 0$ such that for all $n \geq 2$, $|C \cap \Sigma^{(n)}| \geq 2^{n^\epsilon}$. Combining the construction in the previous proof with techniques from [3], we obtain the following corollary.

Corollary 5.12 *For every* DEXT*-complete set B there exist an exponentially dense set of strings C and a constant c such that for every polynomial t and almost all $x \in C$,*

$$\mathrm{ic}^t(x : B) \geq K^{\exp'}(x) - 2\log K^{\exp'}(x) - c,$$

where $\exp'(n) = cn2^{2n} + c$.

Proof. It follows by Proposition 3.3 (ii) that the set constructed in the previous proof is bi-immune. In fact, the diagonalization can easily be interleaved with a construction from [3] to obtain a set that is *strongly* bi-immune, a condition implying that every $\leq^p_m$-reduction from A to any other set is one-to-one almost everywhere. Let B be any DEXT-complete set, and let f be a reduction from A to B. Then f is almost everywhere one-to-one, and consequently the set $C = f(\Sigma^*)$ is exponentially dense. Furthermore, we may assume that f is length-increasing, because all DEXT-complete sets are related by length-increasing reductions [4, 30], honestly paddable DEXT-complete sets exist, and reductions to honestly paddable sets can always be made length-increasing.

By Proposition 3.5, there is a constant c_1 such that for almost all $x \in \Sigma^*$, and hence for almost all $f(x) \in C$,

$$\mathrm{ic}^{\exp}(x : A) \leq \mathrm{ic}^t(f(x) : B) + c_1. \qquad (6)$$

By the properties of A, on the other hand, there is a constant c_2 such that for all x,

$$\mathrm{ic}^{\exp}(x : A) \geq K^{\exp''}(x) - 2\log K^{\exp''}(x) - c_2, \qquad (7)$$

where $\exp''(n) = c_2 n2^{2n} + c_2$.

Let then p be a program computing the reduction $f = f_p$ in time bounded by a nondecreasing polynomial r. Denote $c_3 = |p| + 2\log|p| + \gamma$. Given any string $x \in \Sigma^*$, let q be a minimal size program for producing x in time $\exp''(|x|)$. Then the program $p \circ q$ produces $f(x) = y$, and we obtain the following size and time bounds:

$$\begin{array}{rcl} |p \circ q| & \leq & |p| + |q| + 2\log|p| + \gamma \\ & = & |q| + c_3 \\ & = & K^{\exp''}(|x|) + c_3, \\ \mathrm{time}_{p\circ q}(\lambda) & \leq & \exp''(|x|) + r(|x|) + \gamma(|q| + c_3) \\ & \leq & \exp''(|y|) + r(|y|) + \mathrm{const} \cdot |y| \\ & \leq & c_4 \exp''(|y|), \end{array}$$

for some constant c_4. Let us denote $c = c_4 c_2$ and $\exp'(n) = cn2^{2n} + c$. Because f is almost everywhere one-to-one, we see that for almost all $f(x) \in C$,

$$K^{\exp'}(f(x)) \leq K^{\exp''}(x) + c_3. \tag{8}$$

Combining inequalities (6), (7), and (8), and observing that the function $k - 2\log k$ is monotonically increasing for $k \geq 4$, we then obtain the result that for any constant $c \geq c_1 + c_2 + \max\{c_3, 6\}$ and for almost all $y = f(x) \in C$,

$$\mathrm{ic}^t(y : B) \geq K^{\exp'}(y) - 2\log K^{\exp'}(y) - c. \ \Box$$

6 Conclusion and Further Research

We have introduced a program-size based measure for the complexity of individual instances of a decision problem, and studied the properties of this new notion. The most fundamental questions here concern the existence of instances with high instance complexity, relative to their Kolmogorov complexity. We are putting forth an "instance complexity conjecture", which attempts to formalize the intuitive idea that problems are hard if and only if they have infinitely many intrinsically hard instances. Formally, the conjecture states that if a set A is not in the class DTIME(t), then for infinitely many strings x, the t-bounded instance complexity of x with respect to A is within a constant of the t'-bounded Kolmogorov complexity of x, where $t' = O(t \log t)$.

The results in Section 5 of this paper provide support for this conjecture, and come fairly close to proving it in the case of many natural intractable sets. Obviously, any further results on the conjecture would be extremely interesting — including any results pointing in the opposite direction.

From past experience, resolving the conjecture should be within reach in the limiting, recursive case. Precisely, one would like to prove or disprove the following: for any recursively enumerable, nonrecursive set A, there exist a constant c and infinitely many strings x such that

$$\mathrm{ic}(x : A) \geq K(x) - c,$$

where ic and K denote the time-unbounded versions of instance complexity and Kolmogorov complexity, respectively. Surprisingly, even this seems to be a nontrivial problem.

At present we have no lower bounds on the absolute instance complexity of the "hard instances" provided by the results in Section 5. The construction in the proof of Theorem 5.5 seems to suggest that at least a bound of $\Omega(\log\log n)$ on the complexity of the produced instances could be achieved, but we have not been able to establish this conclusively. Optimally, one might even hope to match the $\Omega(\log n)$ lower bound on absolute instance complexities for NP-hard sets from Section 4.

A Note on Recent Work

Since this paper was completed, Arvind et al. [1] have characterized our class IC[log, poly] as consisting of exactly those sets that can be both conjunctively and disjunctively reduced to tally sets, and proved that the class is downward closed under $\leq_{btt}^{p}$-reductions. These results then yield easy "structural complexity" proofs of our Theorem 4.1 and its corollaries. In another development, Ko [17] has proved that if one-way functions that are secure against polynomial size circuits exist, then any NP-hard set will in fact have a nonsparse set of instances witnessing its noninclusion in the class IC[log, poly]. This result is obtained as a corollary to results on the instance complexities of random and pseudorandom sets.

Acknowledgments

The authors wish to thank Ronald Book for hosting a June 1985 workshop at the University of California, Santa Barbara, where many of the initial results in this work were obtained. The third author would also like to thank Thomas Thierauf for several comments on an earlier version of this paper.

References

[1] Arvind, V., Han, Y., Hemachandra, L., Köbler, J., Mundhenk, M., Schöning, U., Thierauf, T., Lozano, A., Ogiwara, M., and Silvestri, R. Reductions to sets of low information content. In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming* (Vienna, July). *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1992.

[2] Balcázar, J. L., Díaz, J., and Gabarró, J. *Structural Complexity I.* Springer-Verlag, Berlin Heidelberg, 1988.

[3] Balcázar, J. L., and Schöning, U. Bi-immune sets for complexity classes. *Math. Systems Theory 18* (1985), 1–10.

[4] Berman, L. *Polynomial reducibilities and complete sets.* Ph.D. Thesis, Cornell Univ., 1977.

[5] Book, R.V. Tally languages and complexity classes. *Info. Control 26* (1974), 186–193.

[6] Book, R., Du, D.-Z., and Russo, D. A. On polynomial and generalized complexity cores. In *Proceedings of the 3rd Annual Symposium on Structure in Complexity Theory* (Washington, D.C., June). IEEE, New York, N.Y., 1988, pp. 236–250.

[7] Chaitin, G. J. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. Assoc. Comput. Mach. 16* (1969), 407–422. Reprinted in Chaitin, G. J. *Information Randomness & Incompleteness — Papers on Algorithmic Information Theory.* World Scientific, Singapore, 1987, pp. 256–272.

[8] Even, S., Selman, A. L., and Yacobi, Y. Hard core theorems for complexity classes. *J. Assoc. Comput. Mach. 32* (1985), 205–217.

[9] Flajolet, P., and Steyaert, J. M. On sets having only hard subsets. In *Proceedings of the 2nd International Colloquium on Automata, Languages, and Programming* (Saarbrücken, July). *Lecture Notes in Computer Science 14*. Springer-Verlag, Berlin, 1974, pp. 446–457.

[10] Hartmanis, J. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th Annual Symposium on the Foundations of Computer Science* (Tucson, Az., November). IEEE, New York, N.Y., 1983, pp. 439–445.

[11] Hartmanis, J. On sparse sets in NP. *Info. Proc. Letters 16* (1983), 55–60.

[12] Hopcroft, J. E., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Ma., 1979.

[13] Jazayeri, M., Ogden, W. F., and Rounds, W. C. The intrinsically exponential complexity of the circularity problem for attribute grammars. *Comm. Assoc. Comput. Mach. 18* (1975), 697–706.

[14] Karp, R. M., and Lipton, R. J. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing* (Los Angeles, Ca., April). ACM, New York, N.Y., 1980, pp. 302–309.

[15] Ko, K.-I. On the notion of infinite pseudorandom sequences. *Theoret. Comput. Sci. 48* (1986), 9–33.

[16] Ko, K.-I. Non-levelable and immune sets in the accepting density hierarchy for NP. *Math. Systems Theory 18* (1985), 189–205.

[17] Ko, K.-I. A note on the instance complexity of pseudorandom sets. In *Proceedings of the 7th Annual Symposium on Structure in Complexity Theory* (Boston, Ma., June). IEEE, New York, N.Y., 1992.

[18] Kolmogorov, A. N. Three approaches to the quantitative definition of information. *Prob. Info. Transmission 1* (1965), 1–7.

[19] Li, M., and Vitányi, P. M. B. Kolmogorov complexity and its applications. In *Handbook of Theoretical Computer Science. Vol. A: Algorithms and Complexity.* J. van Leeuwen, Editor. Elsevier, Amsterdam, 1990, pp. 187–254.

[20] Loveland, D. W. A variant of the Kolmogorov concept of complexity. *Info. Control 15* (1969), 510–526.

[21] Lynch, N. On reducibility to complex or sparse sets. *J. Assoc. Comput. Mach. 22* (1975), 341–345.

[22] Meyer, A. M., and McCreight, E. M. Computationally complex and pseudorandom zero-one valued functions. In *Theory of Machines and Computations.* Z. Kohavi and A. Paz, Eds. (Haifa, August). Academic Press, New York/London, 1971, pp. 19–42.

[23] Meyer, A. M., and Paterson, M. P. With what frequency are apparently intractable problems difficult? Tech. Rep. TM–126, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Ma., 1979.

[24] Orponen, P. A classification of complexity core lattices. *Theoret. Comput. Sci. 47* (1986), 121–130.

[25] Orponen, P., Russo, D. A., and Schöning, U. Optimal approximations and polynomially levelable sets. *SIAM J. Comput. 15* (1986), 399–408.

[26] Orponen, P., and Schöning, U. The density and complexity of polynomial cores for intractable sets. *Info. Control 70* (1986), 54–68.

[27] Rogers, H., Jr. *Theory of Recursive Functions and Effective Computability.* McGraw–Hill, New York, N.Y., 1967.

[28] Schnorr, C.P. Optimal algorithms for self-reducible problems. In *Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming* (Edinburgh, July). Edinburgh Univ. Press, Edinburgh, 1976, pp. 322–337.

[29] Sipser, M. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing* (Boston, Ma., April). ACM, New York, N.Y., 1983, pp. 330–335.

[30] Watanabe, O. On one-one polynomial time equivalence relations. *Theoret. Comput. Sci. 38* (1985), 157–165.

[31] Wilber, R.E. Randomness and the density of hard problems. In *Proceedings of the 24th Annual Symposium on the Foundations of Computer Science* (Tucson, Az., November). IEEE, New York, N.Y., 1983, pp. 335–342.