Generation complexity versus distinction complexity

Rupert Hölzl and Wolfgang Merkle

Institut für Informatik,
Ruprecht-Karls-Universität,
Heidelberg, Germany,
{hoelzl|merkle}@math.uni-heidelberg.de

Abstract. Among the several notions of resource-bounded Kolmogorov complexity that suggest themselves, the following one due to Levin [Le] has probably received most attention in the literature. With some appropriate universal machine U understood, let the Kolmogorov complexity of a word w be the minimum of $|d| + \log t$ over all pairs of a word d and a natural number t such that U takes time t to check that d determines w. One then differentiates between generation complexity and distinction complexity [A,Sip], where the former asks for a program d such that w can actually be computed from d, whereas the latter asks for a program d that distinguishes w from other words in the sense that given d and any word u, one can effectively check whether u is equal to w.
Allender et al. [A] consider a notion of solvability for nondeterministic computations that for a given resource-bounded model of computation amounts to require that for any nondeterministic machine N there is a deterministic machine that exhibits the same acceptance behavior as N on all inputs for which the number of accepting paths of N is not too large. They demonstrate that nondeterminism is solvable for computations restricted to polynomially exponential time if and only if for any word the generation complexity is at most polynomial in the distinction complexity. We extend their work and a related result by Fortnow and Kummer [FK] as follows. First, nondeterminism is solvable for linearly exponential time bounds if and only if generation complexity is at most linear in distinction complexity. Second, nondeterminism is solvable for polynomial time bounds if and only if the conditional generation complexity of a word w given a word y is at most linear in the conditional distinction complexity of w given y; hence, in particular, the latter condition implies that **P** is equal to **UP**. Finally, in the setting of space bounds it holds unconditionally that generation complexity is at most linear in distinction complexity.

In general, the Kolmogorov complexity of a word w is the length $|d|$ of a shortest program d such that d determines w effectively. In a setting of unbounded computations, this approach leads canonically to the usual notion of plain Kolmogorov complexity and its prefix-free variant. In a setting of resource-bounded

computations though, there are several notions of Kolmogorov complexity that are in some sense natural – and none of them is considered canonical.

A straight-forward approach is to cap the execution time and/or used space by simply not allowing descriptions that take too long or too much space for producing the word we want to describe. This notion has the disadvantage that for a fixed resource-bound there is no canonical notion of universal machine. Another approach, which has received considerable attention in the literature, was introduced by Levin [Le], where, in contrast to the notion just mentioned, arbitrarily long computations are allowed, but a large running time increases in some way the complexity value. More precisely, with some appropriate universal machine U understood, in Levin's model the Kolmogorov complexity of a word d is the minimum of $|d| + \log t$ over all pairs of a word d and a natural number t such that U takes time t to check that w is the word determined by d. As for other notions of resource-bounded Kolmogorov complexity, here one can differentiate between generation complexity and distinction complexity [A,Sip], where the former asks for a program d such that w can actually be computed from d, whereas the latter asks for a program d that distinguishes w from other words in the sense that given d and any word u, one can effectively check whether u is equal to w.

The question of how generation and distinction complexity relate to each other in the setting of Levin's notion of resource-bounded Kolmogorov complexity has been investigated by Allender et al. [A]. They consider a notion of solvability for nondeterministic computations that — for a given resource-bounded model of computation — amounts to require that for any nondeterministic machine N there is a deterministic machine that exhibits the same acceptance behavior as N on all inputs for which the number of accepting paths of N is not too large, e.g., is at most logarithmic in the number of all possible paths. Their main result then asserts that nondeterminism is solvable for computations restricted to polynomially exponential time if and only if for any word the generation complexity is at most polynomial in the distinction complexity.

We extend the work of Allender et al. [A] and a related result by Fortnow and Kummer [FK] as follows. First, nondeterminism is solvable for linearly exponential time bounds if and only if generation complexity is at most linear in distinction complexity. Second, nondeterminism is solvable for polynomial time bounds if and only if the conditional generation complexity of a word w given a word y is at most linear in the conditional distinction complexity of w given y; as a consequence, the latter condition implies in particular that $\mathbf{P}$ is equal to $\mathbf{UP}$. Combining the result on polynomial time bounds with a result by Fortnow and Kummer [FK] about Kolmogorov complexity defined in terms of fixed polynomial time bounds, one obtains that in the model just mentioned conditional generation and distinction complexity are close if and only if they are close in Levin's model. Finally, in the setting of space bounds, more precisely, for complexity measures Ks and KDs that logarithmically count the used space instead of the running time used on a program, it holds unconditionally that generation complexity is at most linear in distinction complexity.

The notion of generation complexity considered below differs from Levin's original notion insofar as one has to generate only single bits of the word to be generated but not the word as a whole. This variant has already been used by Allender et al. [A]; their results mentioned above, as well as the results demonstrated below extend to Levin's original model by almost identical proofs.

For a complexity class **C**, we will refer by **C**-machine to any machine M that uses a model of computation and obeys a time- or space-bound such that M witnesses $\mathrm{L}(M) \in \mathbf{C}$ with respect to the standard definition of **C**. For example, an **NE**-machine is a nondeterministic machine that runs in linearly exponential time.

The individual bits of a word x will be denoted by x_1 to $x_{|x|}$. We fix an appropriate universal machine U that receives as input encoded tuples of words and e.g. (x, y, z) will be encoded by $\tilde{x}01\tilde{y}01\tilde{z}$ where the word $\tilde{u}$ is obtained by doubling every symbol in u, i.e., $\tilde{u} = u_1u_1u_2u_2 \ldots u_{|u|}u_{|u|}$.

Logarithms to base 2 are denoted by log, and often a term of the form $\log t$ will indeed denote the least natural number s such that $t \leq 2^s$.

1 Known results

Definition 1 (Levin [Le], Allender et al. [A]). *Time-bounded generation complexity* $\mathrm{Kt}(.)$ *and distinction complexity* $\mathrm{KDt}(.)$ *are defined by*

$$\mathrm{Kt}(x) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall b \in \{0,1,*\}\colon \forall i \leq |x|+1\colon U(d,i,b) \\ \text{runs for } t \text{ steps and accepts iff } (x*)_i = b \end{array} \right\},$$

$$\mathrm{KDt}(x) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall y \in \Sigma^{|x|}\colon U(d,y) \text{ runs for} \\ t \text{ steps and accepts iff } x = y \end{array} \right\}.$$

Observe that in the definition of Kt-complexity the symbol $*$ has to be generated as an end marker for the word x.

Remark 2. The notion of Kt-complexity introduced in Definition 1 was proposed by Allender et al. in [A] as a variation of Levin's original definition, where the latter requires to generate whole words instead of individual bits. Levin's original definition has the advantage of assuring that for all x, it holds that $\mathrm{KDt}(x) \leq \mathrm{Kt}(x) + \log|x|$.

In connection with Theorem 14 we also use the following conditional complexity notions.

Definition 3. *The conditional time-bounded distinction complexity* $\mathrm{Kt}(.|.)$ *and conditional generation complexity* $\mathrm{KDt}(.|.)$ *are defined by*

$$\mathrm{Kt}(x|y) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{c} \forall b \in \{0,1,*\}\colon \forall i \leq |x|+1\colon U(d,y,i,b) \\ \text{runs for } t \text{ steps and accepts iff } (x*)_i = b \end{array} \right\},$$

$$\mathrm{KDt}(x|y) = \min\left\{ |d| + \log t \;\middle|\; \begin{array}{l} \forall z \in \Sigma^{|x|} \colon U(d,y,z) \text{ runs for} \\ t \text{ steps and accepts iff } z = x \end{array} \right\}.$$

We will shortly review Theorem 17 from Allender et al. [A] before we will state our extensions.

Definition 4 (Allender et al. [A]). *We say that* **FewEXP** search instances are **EXP**-solvable *if, for every* **NEXP***-machine N and every k, there is an* **EXP***-machine M with the property that if N has fewer than $2^{|x|^k}$ accepting paths on input x, then M produces on input x some accepting path as output if there is one.*

We say that **FewEXP** decision instances are **EXP**-solvable *if, for every* **NEXP***-machine N and every k, there is an* **EXP***-machine M with the property that if N has fewer than $2^{|x|^k}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that **FewEXP** decision instances are **EXP**/poly-solvable *if, for every* **NEXP***-machine N and every k, there is an* **EXP***-machine M having access to advice of polynomial length, such that if N has fewer than $2^{|x|^k}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

The notion of solvability can be equivalently characterized in terms of promise problems [CHV,FK]. This will be discussed further in connection with Theorem 17 by Fortnow and Kummer.

Remark 5. Note that by definition **EXP**-solvability of **FewEXP** decision instances implies **FewEXP** = **EXP**, but it is unknown whether the reverse implication holds as well. This is because the definition of **EXP**-solvability does *not* require the considered machines to have a limited number of accepting paths on *all* inputs.

Theorem 6 (Allender et al. [A]). *The following statements are equivalent:*

1 *For all x,* $\mathrm{Kt}(x) \in (\mathrm{KDt}(x))^{\mathrm{O}(1)}$.
2 **FewEXP** *search instances are* **EXP***-solvable.*
3 **FewEXP** *decision instances are* **EXP***-solvable.*
3′ **FewEXP** *decision instances are* **EXP**/poly*-solvable.*
4 *For all $A \in \mathbf{P}$ and for all $y \in A^{=l}$ it holds that*

$$\mathrm{Kt}(y) \in (\log |A^{=l}| + \log l)^{\mathrm{O}(1)}.$$

In words this means, that if generating is "not much more difficult" than distinguishing, then witnesses for certain nondeterministic computations with few witnesses can be found deterministically, and vice versa.

2 Tools

In what follows we will use a corollary of the following result by Buhrman et al., which have also been used by Allender et al. We omit proofs and more detailed discussion due to space considerations.

Lemma 7 (Buhrman et al. [BFL]). *Let n be large enough and let*

$$A := \{x_1, x_2, \ldots, x_{|A|}\} \subseteq \{l, l+1, \ldots, l+n-1\}.$$

Then for all $x_i \in A$ and at least half of the prime numbers $p \leq 4 \cdot |A| \cdot \log^2 n$ it holds for all $j \neq i$ that $x_i \not\equiv x_j \pmod p$.

Corollary 8. *Let $A \subseteq \Sigma^*$, $y \in \Sigma^*$ and $l \in \mathbb{N}$. Let*

$$A_{y,l} := A \cap \{x \mid y \sqsubseteq x \wedge |x| = l\}.$$

Then it holds that

$$\forall y \in A_{y,l} \colon \mathrm{KDt}^{A_{y,l}}(x) \leq 2 \log |A_{y,l}| + \mathrm{O}(\log l)$$

In particular, if there is a machine that on input y, l and x decides in polynomial time whether x is in the set $A_{y,l}$, then

$$\forall x \in A_{y,l} \colon \mathrm{KDt}(x|y) \leq 2 \log |A_{y,l}| + \mathrm{O}(\log l).$$

3 New results

We will now state our variants of Theorem 6 by Allender et al., which are demonstrated by similar proofs.

Definition 9. *We say that* **FewE** search instances are **E**-solvable *if, for every* **NE**-*machine N and every k, there is an* **E**-*machine M with the property that if N has fewer than $2^{k \cdot |x|}$ accepting paths on input x, then M produces on input x some accepting path as output if there is one.*

We say that **FewE** decision instances are **E**-solvable *if, for every* **NE**-*machine N and every k, there is an* **E**-*machine M with the property that if N has fewer than $2^{k \cdot |x|}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that **FewE** decision instances are **E**/lin-solvable *if, for every* **NE**-*machine N and every k, there is an* **E**-*machine M having access to advice of linear length, such that if N has fewer than $2^{k \cdot |x|}$ accepting paths on input x, then M accepts x if and only if N accepts x.*

We say that **UE** decision instances are **E**-solvable *if, for every* **NE**-*machine N and every k, there is an* **E**-*machine M with the property that if N has at most one accepting path on input x, then M accepts x if and only if N accepts x.*

Theorem 10. *The following statements are equivalent:*

1 *For all words* x, $\mathrm{Kt}(x) \in \mathrm{O}(\mathrm{KDt}(x))$.
2 **FewE** *search instances are* **E***-solvable.*
3 **FewE** *decision instances are* **E***-solvable.*
3′ **UE** *decision instances are* **E***-solvable.*
3″ **FewE** *decision instances are* **E**/lin*-solvable.*
4 *For all* $A \in \mathbf{P}$ *it holds that for* $A_{y,l} := A \cap \{x \mid y \sqsubseteq x \wedge |x| = l\}$ *and for all* $x \in A_{y,l}$

$$\mathrm{Kt}(x) \in \mathrm{O}(\log |A_{y,l}| + \log l + |y|).$$

We omit the proof of Theorem 10 due to space constraints.

Remark 11. Theorem 10 remains valid by essentially the same proof when formulated for Levin's original notion of Kt instead of the variant of Allender et al.

Corollary 12. *If for all* x, $\mathrm{Kt}(x) \in \mathrm{O}(\mathrm{KDt}(x))$, *then* $\mathbf{UE} = \mathbf{E}$.

Proof. According to the theorem, the assumption implies that **UE** decision instances are **E**-solvable. Since a language in **UE** contains only such instances, the claim follows. □

The equivalence stated in Theorem 10 can be extended to the setting of polynomial time bounds when considering conditional complexities.

Definition 13. *We say that* **FewP** search instances are **P**-solvable *if, for every* **NP** *machine* N *and every* k *there is a* **P** *machine* M *with the property that if* N *has fewer than* $|x|^k$ *accepting paths on input* x, *then* M *produces on input* x *some accepting path as output if there is one.*

We say that **FewP** decision instances are **P**-solvable *if, for every* **NP** *machine* N *and every* k *there is a* **P** *machine* M *with the property that if* N *has fewer than* $|x|^k$ *accepting paths on input* x, *then* M *accepts* x *if and only if* N *accepts* x.

We say that **UP** decision instances are **P**-solvable *if, for every* **NP***-machine* N *and every* k, *there is a* **P***-machine* M *with the property that if* N *has at most one accepting path on input* x, *then* M *accepts* x *if and only if* N *accepts* x.

Theorem 14. *The following statements are equivalent:*

1 *For all words* x *and* y, $\mathrm{Kt}(x|y) \in \mathrm{O}(\mathrm{KDt}(x|y))$.
2 **FewP** *search instances are* **P***-solvable.*
3 **FewP** *decision instances are* **P***-solvable.*
3′ **UP** *decision instances are* **P***-solvable.*
4 *For all* $A \in \mathbf{P}$ *it holds that for* $A_{y,l} := A \cap \{x \mid y \sqsubseteq x \wedge |x| = l\}$ *and for all* $x \in A_{y,l}$

$$\mathrm{Kt}(x|y) \in \mathrm{O}(\log |A_{y,l}| + \log l).$$

Proof. $(1 \Rightarrow 4)$**:** We have access to y through the conditioning. If we also have access to l, we can decide membership of x in $A_{y,l}$ in polynomial time. To do this, we first check whether $y \sqsubseteq x$ and whether y has the correct length l. If yes, compute the value $A(x) = A_{y,l}(x)$ using the fact that $A \in \mathbf{P}$. Corollary 8 then yields

$$\mathrm{KDt}(x|y) \leq 2 \log |A_{y,l}| + \mathrm{O}(\log l).$$

Using assumption 1 the claim follows.

$(4 \Rightarrow 2)$**:** Let N be any nondeterministic machine running in polynomial time n^k, where we can assume that N branches binarily. Let L denote $\mathrm{L}(N)$. Let

$$D := \{yx \mid x \in \{0,1\}^{|y|^k} \text{codes an accepting computation of } N \text{ on } y\}.$$

Obviously, $D \in \mathbf{P}$. Now fix any y such that M on input y has at most $|y|^k$ accepting paths. Then the set $D_y := D \cap \{yx \mid |x| = |y|^k\}$ contains at most $|y|^k$ words and by assumption 4 it follows that

$$\begin{aligned} \forall yx \in D_y \colon \mathrm{Kt}(yx|y) &\in \mathrm{O}(\log |D_y| + \log(|y| + |y|^k)) \\ &= \mathrm{O}(\log(|y|)) \end{aligned}$$

So, in order to find an accepting path of M on input y, if there is one, it suffices to search through all words x with $\mathrm{Kt}(x|y) \leq \mathrm{O}(\log |y|)$. This can be done in polynomial time, so that $L \in \mathbf{P}$ as was to be shown. Note that it causes no problems that we have to deal with conditional complexity here. This is because when we are searching for an accepting path x for a word y we obviously have access to y.

$(2 \Rightarrow 3)$**:** This is trivial.

$(3 \Rightarrow 3')$**:** This is trivial.

$(3' \Rightarrow 1)$**:** Let us assume that we have a KDt-description d, finite conditioning information y and a described word x such that the universal machine U accepts the triple (d, y, x) in t_{KDt} steps. Assuming that this is the optimal description for x we would have $\mathrm{KDt}(x|y) = |d| + \log t_{\mathrm{KDt}}$. Since we can in t_{KDt} steps only access the first t_{KDt} bits of y (with the encoding of tupels introduced with the universal machine U) we can use $y[1..t_{\mathrm{KDt}}]$ instead of y for the rest of the proof and can therefore w.l.o.g. assume $|y| < t_{\mathrm{KDt}}$. By a similar argument we can assume $|d| < t_{\mathrm{KDt}}$.

Consider a variant $U_{\mathcal{UP}}$ of the universal machine U where $U_{\mathcal{UP}}$ is given inputs of the form $(\hat{d}, \hat{y}, 1^{\hat{t}}, \hat{\imath}, \hat{b}, \hat{n})$. On any such input, if $\hat{n} > \hat{t}$, then reject. Otherwise guess a word $\hat{x} \in \{0,1\}^{\hat{n}}$ and check whether $\hat{x}_{\hat{\imath}} = \hat{b}$. If yes, $U_{\mathcal{UP}}$ behaves for $\hat{t}$ steps like U on input $(\hat{d}, \hat{y}, \hat{x})$, that is $U_{\mathcal{UP}}$ accepts iff $U(\hat{d}, \hat{y}, \hat{x})$ accepts in these $\hat{t}$ steps.

For our fixed triple (d, y, x) this computation takes $t_{\mathcal{UP}}$ steps, with $t_{\mathcal{UP}} \in \Theta(|x| + t_{\mathrm{KDt}})$. Note that the running time of the simulation is coded unarily into its input. Let's call the coded number t_{coded}. So we have $t_{\mathrm{KDt}} = t_{\mathrm{coded}}$. The execution of a distinguishing description for x on U takes at least $|x|$ steps (otherwise x could not even be read completely, and therefore it could not be correctly distinguished). So we have $t_{\mathcal{UP}} \in \Theta(t_{\mathrm{KDt}})$.

This computation is now a nondeterministic one that already correctly recognizes the given bit of x. Because no part of the input pentuple P is longer than t_{coded}, we have that the program has length $\Theta(t_{\text{coded}}) = \Theta(t_{\text{KDt}})$ and that therefore the running time $t_{\mathcal{UP}}$ of the nondeterministic computation is in $\Theta(|P|)$.
Since d is a *distinguishing* description for the word $x \in \{0,1\}^n$, for all i on input $(d, y, 1^{t_{\text{coded}}}, i, x_i, |x|)$ there is a unique accepting path of $U_{\mathcal{UP}}$ and none on input $(d, y, 1^{t_{\text{coded}}}, i, \bar{x}_i, |x|)$. By assumption $3'$ there is a deterministic machine M that for all such inputs has the same acceptance and rejection behaviour as N and works in some fixed polynomial time bound.
The input for M together with an encoding of M is a generating program for x. It only remains to prove that this program is small enough and computes fast enough, compared to the KDt-program. This then implies $\text{Kt}(x|y) \leq \text{O}(\text{KDt}(x|y))$, as desired.
Let us first inspect the program length. The input for M consists of t_{coded}, $|x|$ (both encoded in binary), $\ulcorner M \urcorner$, d. Since t_{KDt} counted logarithmically for KDt we have $\log t_{\text{coded}} \leq \text{KDt}(x|y)$. $\text{KDt}(x|y)$ is always greater than $\log |x|$, for the same argument as above. One fixed M works for all appropriate inputs and its encoding therefore has constant length. Obviously, $d \leq \text{KDt}(x|y)$. Furthermore, y is given as part of the input to M, but does not add to $\text{Kt}(x|y)$.
Let us now inspect the running time of the code. The nondeterministic machine used a running time in $\Theta(|P|)$. After the conversion to a deterministic procedure we have by assumption a running time of $(|P|)^{\text{O}(1)}$. In other words: A polynomial overhead might have been introduced relative to the nondeterministic running time $t_{\mathcal{UP}}$. Therefore we have $t_{\mathbf{P}} \in \text{O}((t_{\mathcal{UP}})^c) = \text{O}(t^c_{\text{KDt}})$, hence $\log(t_{\mathbf{P}}) \in \text{O}(\log t_{\text{KDt}})$.
All this, together with the fact that running time counts logarithmically, results in the required inequality $\text{Kt}(x|y) \leq \text{O}(\text{KDt}(x|y))$. □

Remark 15. For the same reason as in Remark 11, this proof would also work if we considered Levin's original definition of Kt.

Corollary 16. *If for all x and y, $\text{Kt}(x|y) \in \text{O}(\text{KDt}(x|y))$, then $\mathbf{UP} = \mathbf{P}$.*

Proof. According to the theorem, the assumption implies that **UP** decision instances are **P**-solvable. Since a language in **UP** contains only such instances, the claim follows. □

Fortnow and Kummer [FK, Theorem 24] proved an equivalence related to Theorem 14 in the setting of the "traditional" polynomially time-bounded Kolmogorov complexities C^t and CD^t [LiV, Chapter 7] where for example

$$\text{C}^t(x) = \min\{|d| \mid U(d) \text{ runs on input } x \text{ for } t(|x|) \text{ steps and outputs } x\}.$$

Theorem 17 (Fortnow, Kummer). *The following two statements are equivalent:*

1. **UP** *decision instances are* **P***-solvable.*
2. *For any polynomial t there are a polynomial t' and a constant $c \in \mathbb{N}$ such that for all x and y it holds that $C^{t'}(y|x) \leq CD^t(y|x) + c$.*

Remark 18. In fact Fortnow and Kummer formulated their equivalence in terms of promise problems. Instead of the first statement in the theorem they used the assertion that the promise problem $(1\mathrm{SAT}, \mathrm{SAT})$ is in **P**, where for a promise problem (Q, R) to be in **P** means that there is a **P**-machine that accepts all $x \in Q \cap R$ and rejects all $x \in \Sigma^* - R$.

Their formulation of the first statement is indeed equivalent to the one used above, because $(1\mathrm{SAT}, \mathrm{SAT})$ is complete for $\mathcal{UP}$, as witnessed by a parsimonious version of Cook's Theorem due to Simon [Sim, Theorem 4.1].

The following corollary is immediate from Theorems 14 and 17.

Corollary 19. *The following two statements are equivalent.*

1. *For all x and y, $\mathrm{Kt}(x|y) \in \mathrm{O}(\mathrm{KDt}(x|y))$.*
2. *For any polynomial t there is a polynomial t' and a constant $c \in \mathbb{N}$ such that for all x and y it holds that $C^{t'}(y|x) \leq CD^t(y|x) + c$.*

In analogy to the time-bounded case one can define the following two notions of space-bounded Kolmogorov complexity.

Definition 20. *The space-bounded distinction complexity* Ks *and generation complexity* KDs *are defined by*

$$\mathrm{Ks}(x) = \min\left\{ |d| + \log s \;\middle|\; \begin{array}{c} \forall b \in \{0,1,*\} \colon \forall i \leq |x|+1 \colon U(d,i,b) \\ \text{runs in space } s \text{ and accepts iff } (x*)_i = b \end{array} \right\},$$

$$\mathrm{KDs}(x) = \min\left\{ |d| + \log \max(s, |x|) \;\middle|\; \begin{array}{c} \forall y \in \Sigma^{|x|} \colon U(d,y) \text{ runs in} \\ \text{space } s \text{ and accepts iff } x = y \end{array} \right\}.$$

Here U is a machine with a two-way read-only input tape, where only the space on the work tapes is counted.

Remark 21. For the definition of KDs it is relevant how the candidate y is provided to U and if the space for y is counted. Here we chose to *do* count the space for y which accounts for the term $\max|x|$ in the definition of KDs. This then implies the inequality $\log|x| \leq \mathrm{KDs}(x)$, which is analogous to the corresponding statement for KDt and will be used in the proof of Theorem 22.

Theorem 22. *For almost all x, it holds that* $\mathrm{Ks}(x) \leq 5 \cdot \mathrm{KDs}(x)$.

Proof. Let N be a nondeterministic machine which on input (d, s, i, b, n) guesses a word $y \in \{0,1\}^n$, simulates the computation of $U(d, y)$ while limiting the used space to s, and then accepts iff $y_i = b$ and $U(d, y)$ accepts. In particular, if d is a

distinguishing description for a word $x \in \{0,1\}^n$, then for all sufficiently large s and for all $i \leq n$ there is an accepting path of N on input $(d, s, i, x_i, |x|)$ but none on $(d, s, i, \bar{x_i}, |x|)$.

By the Theorem of Savitch there is a deterministic machine M that has the same acceptance behavior as N and uses space at most s^2; observe in this connection that s is specified in the input of N and M, hence doesn't have to be computed by M.

Given a word x, fix a pair d and s such that d is a distinguishing program for x, it holds that $|d| + \log s \leq \mathrm{KDs}(x)$, and U uses space at most s on input (d, x). The specification of d, s, $|x|$ and M therefore constitutes a Ks-program for x which runs in space s^2. By choice of d and s we have

$$|d| + \log s + \log |x| \leq 2\mathrm{KDs}(x).$$

Furthermore, the space s^2 used in the computation of M counts only logarithmically, where $2 \cdot \log s \leq 2 \cdot \mathrm{KDs}(x)$. Taking into account that M has to be specified and that some additional information is needed to separate the components of the Ks-program for x, we obtain $\mathrm{Ks}(x) \leq 5 \cdot \mathrm{KDs}(x)$ for all sufficiently large x. □

References

[A] E. Allender, M. Koucký, D. Ronneburger, S. Roy. Derandomization and distinguishing complexity. *Proc. 18th Annual IEEE Conference on Computational Complexity*, pp. 209-220, IEEE Computer Society, 2003.

[BFL] H. Buhrman, L. Fortnow, S. Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887-905, 2002.

[CHV] J. Cai, L.A. Hemachandra, J. Vyskoč. Promises and fault-tolerant database access. *Complexity Theory. Edited by K. Ambos-Spies, S. Homer, U. Schöning*, pp. 227-244, Cambridge University Press, 1993.

[FK] L. Fortnow, M. Kummer. On resource-bounded instance complexity. *Theoretical Computer Science*, 161:123-140, 1996.

[Le] L.A. Levin. Randomness conservation inequalities: Information and independence in mathematical theories. *Information and Control*, 61:15-37, 1984.

[LiV] M. Li, P. Vitányi. An Introduction to Kolmogorov Complexity and Its Applications. *Springer*, 1997.

[Sim] J. Simon. On some central problems in computational complexity. *Technical Report TR75-224*, Cornell University, 1975.

[Sip] M. Sipser. A complexity theoretic approach to randomness. *Proc. 15th ACM Symp. Theory Comput.*, pp. 330-335, ACM, 1983.