Kolmogorov Complexity and Derandomization

by

Detlef Ronneburger

A Dissertation submitted to the

Graduate School of New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy,

Graduate Program in Computer Science

Written under the direction of

Dr. Eric W. Allender

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2004

ABSTRACT OF THE DISSERTATION

Kolmogorov Complexity and Derandomization

by DETLEF RONNEBURGER

Dissertation Director:
Eric Allender

Kolmogorov complexity is a measure that describes the compressibility of a string. Strings with low complexity contain a lot of redundancy, while strings with high Kolmogorov complexity seem to lack any kind of pattern. For instance, a string such as 5555 5555 5555 5555 5555 has low complexity, while a sequence such as 1732 7356 2748 7621 6552 would have high complexity.

This thesis studies different notions of resource-bounded Kolmogorov complexity. In particular it studies Levin's Kt complexity and measures Kμ that are defined in a similar manner. Levin defined his measure as $\mathrm{Kt}(x) = \min\{\, |d| + \log t \;\mid\; U(d) = x \text{ in } t \text{ steps} \,\}$ where U is a universal Turing machine. It is shown that, contrary to common intuition, the measures Kμ behave differently from the resource-unbounded Kolmogorov complexity, even for generous resource bounds. In particular it is argued that a property called *Symmetry of Information* does not hold for some of these measures Kμ.

One of the main results of this thesis addresses the question of the complexity of computing the measure $\mathrm{Kt}(x)$ for a given string x. It can be computed in exponential

time, but no meaningful lower bound is known. However, it is shown that it is complete for exponential time under efficient, non-uniform reductions (i.e., reductions computable in $\mathsf{P}/_{\mathsf{poly}}$) as well as nondeterministic polynomial time reductions (i.e., reductions computable in NP). Further completeness results of other complexity measures for different complexity classes are obtained as well. These results are of interest as the problem of computing the Kolmogorov complexity of a string is not a typical complete problem. Most problems that are complete for a complexity classes have a clear combinatorial structure representative of the complexity class they reside in. However, the problems studied seem to lack that property.

This thesis also studies the relation between (a) the ability of sets in certain complexity classes to avoid simple strings and (b) the inclusion relation between different complexity classes. For instance, it is shown that every set in P contains simple strings, if and only if $\mathsf{NEXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$.

Acknowledgments

I would like to express my gratitude to all parties that helped me to complete this dissertation. In particular, I need to explicitly mention the following people. Without their contribution I would not be where I am now.

First and foremost, I need to address a deep thank-you to my adviser Eric Allender. Throughout the years he always managed to find that narrow line of providing guidance without stifling individual creativity. He helped me profoundly to gain an understanding and appreciation of complexity theory. Through patient and thorough discussion of my many (so often unsuccessful) proofs, he taught me the precision and rigor that is required for sound scientific work. Eric has been very helpful with and understanding of my sometimes not so easy family circumstances. His dedication to his students, his nurturing of a student's talents and his efforts to help a student overcome weaker points, will always stay with me and have a profound impact in my own academic career.

I am very thankful for my coauthors who strongly contributed to obtain the results presented in this thesis. Many of the results were initially presented in *Power from Random Strings*, coauthored with Eric Allender, Harry Buhrman, Michal Koucký, and Dieter van Melkebeek and presented at FOCS, 2002. Other results were initially presented at the Complexity Conference 2003 in the paper *Derandomization and Distinguishing Complexity*, which was coauthored with Eric Allender, Michal Koucký and Sambuddha Roy. Further I would like to thank Rahul Santhanam and Philippe Moser for the collaboration on some ideas that never made it into a publication.

I would like to express my gratitude to Richard Beigel, Michael Saks, and Mario Szegedy who volunteered to serve on my dissertation committee. In addition I would like thank the members of the computer science department at Rutgers for providing a creative and stimulating research environment. In particular I should mention

Mike Saks, Bill Steiger, Mario Szegedy and Endré Szemeredi, who helped me in their courses to gain a better understanding of different aspects in theoretical computer science. I wish I had utilized their insight and wisdom more. Besides Michal Koucký and Sambuddha Roy whose collaboration resulted in publication, I also would like to thank the other participants of our informal complexity theory seminar, namely Navin Goyal, Xiaomin Chen, Miguel Mosteiro, Bin Tian, and Venkatesh Srinivasan for stimulating discussions. Finally I would like to thank Stephen Max and Marcello Mydlarz for always being available for discussion about topics other than complexity theory. I should also mention my sincere thanks to the secretaries in the department. They have helped tremendously in situations to many to count. In this context I should mention in particular Diana Creighton, Shirley Hinds, and Val Rolfe.

Finally, I have to extend my thanks to my family for their patience and support through the past six years. In particular I must mention my wife Lisa who has taken on many additional stressful and demanding responsibilities so I could focus on my work. She provided the strong emotional support that I needed to finish this thesis.

Part of my work was supported by NSF grants CCR-9734918 and CCR-0104823.

Contents

1 Introduction

The fundamental goal of computational complexity is to understand what resources are inherently required to solve given problems. The first approach of considering the algorithmic complexity of a problem by studying the required resources with regard to the size of the problem instance was presented in [HS65]. It became evident that the vast majority of problems considered naturally fall into a relatively small number of categories, with all problems within a category being essentially equivalent with regard to their computational complexity. So for instance, there is an extremely large number of NP-complete problems ([GJ79]). Many of them turn out to be computationally equivalent by very strict measures. (More precisely they are equivalent under AC^0-reductions.) By classifying problems into these categories, insight about the combinatorial properties of one problem can often be translated into a better understanding of another problem in the same complexity class.

Besides just measuring time and space as resource bounds in order to understand the inherent hardness of problems, complexity theory also considers different computational frameworks and the question of whether any of these models are (provably ?) more powerful than others. For instance, in the context of computability theory it is well known that any deterministic Turing machine can compute anything that can be computed by a nondeterministic Turing machine. In fact, anything that can be computed by any formal model of an algorithm can be computed by a Turing machine. However, in many cases it is not at clear if a formally more powerful model in fact yields more efficient computation. This is essentially the famous P vs. NP question: "Can everything that can be computed in polynomial time by a *nondeterministic* Turing machine be computed by a polynomial time *deterministic* Turing machine?"

In addition to deterministic and nondeterministic Turing machines, complexity theory has considered a large variety of computational models. People have considered

the computational complexity of problems with algorithms implemented as Boolean circuits, branching programs and even on quantum machines.

One notion of computation that has received a lot of attention over the past 20 years is the model of randomized algorithms. For many problems the ability to access random bits makes it easier to design algorithms - or to prove correctness of an algorithm. For other problems the known randomized algorithms are significantly more efficient than the known deterministic algorithms. The underlying question in this context is: Does randomness make computation inherently more powerful?

In practice there are many randomized algorithms being used - on deterministic computers. Instead of using truly random bits, the algorithm just uses the output of a deterministic program: a pseudo-random generator. While heuristically speaking these generators seems to perform well - the derandomized algorithms seems to run efficiently and yield correct results with very high probability - from a theoretical perspective it is very important to know whether randomized algorithms can generally be efficiently made deterministic. This would show that randomness does not inherently boost the power of computation. The research that has addressed this question is usually refered to as the *study of derandomization*.

1.1 Derandomization

The first approaches toward building a pseudo-random generator, a PRG (that is, a deterministic program that receives a relatively short seed as input and whose output cannot be efficiently distinguished from truly random bits), go back to work by Blum and Micali [BM84], and Yao [Yao82]. Yao showed how to build such a generator based on a a one-way permutation - a permutation that is easy to compute but hard to invert. In subsequent work ([ACGS84, Lev85, GKL90, HILL99]) this approach was generalized to building pseudo-random generators built on arbitrary one-way functions.

In order to derandomize a probabilistic algorithm, one can run the generator on every possible short seed and then determine the behavior of the randomized algorithm with respect to each generated sequence of pseudo-random bits. Given a secure generator, the algorithm is not able to discern that the given input is not truly random and it will show the same probability of acceptance when measured over the distribution of pseudo-random strings as when measured over the uniform distribution. Thus the derandomized algorithm just needs to output the Boolean result that has occurred with the higher probability over all considered pseudo-random sequences.

Note that this application of a pseudo-random generator involves a run-time that is exponential in the length of the seed, as the generator is run on every possible seed. Thus it is not necessary that the run-time of the generator is polynomial in terms of the length of the input. This gives rise to a whole new family of pseudo-random generators based on the work presented in Nisan and Wigderson's seminal paper [NW94]. The idea in their approach is to use a computationally hard function f (in this case a function computable in exponential time that cannot be computed by small circuits) in a combinatorial construction to "mix-up" the random bits of the provided seed to obtain a long pseudo-random output. The crucial part of this construction is the fact that any algorithm that can distinguish the output of this generator from truly random bits can be used to build a small circuit to compute the hard function f. Thus their result either implies a non-uniform lower bound for functions computable in exponential time, or a deterministic upper bound bound for randomized algorithms. Results of this nature are called hardness-randomness trade-offs.

Based on this Nisan-Wigderson generator, several improved constructions were considered ([BFNW93, IW97, IW98, STV01, KvM02]). They obtain hardness-randomness trade-offs with stronger parameters or in slightly different settings. We will make heavy use of these constructions to obtain new completeness results for different complexity classes.

1.2 Kolmogorov Complexity

The area of derandomization tries to understand what computational power might or might not be gained by using randomness in computation. In this process several results have established connections between the hardness of a particular function and resources required to solve randomized algorithms. In this thesis we will establish that there is a close connection between the hardness of a function and the randomness of the particular characteristic string associated with that function. In the same context we will address another perspective of how randomness might help in computation. We will ask the question: "How hard is it to recognize a string as random and how much computational power can be gained from this ability?" In order to do this, it is necessary to formally and precisely capture the notion of what it means for a string to be random.

There have been several attempts at defining criteria that characterize an infinite sequence as random [vM39, Mar66b]. One of the first approaches to consider some notion of randomness in the context of finite strings is due to Shannon [Sha48]. He considered the notion of entropy as the amount of randomness contained in the distribution of a random variable over a finite set of strings.

However, we will consider the randomness of individual strings without the context of a probability distribution. The complexity measure that we will use, Kolmogorov or K-complexity, initially goes back to Kolmogorov ([Kol68]), Chaitin ([Cha66, Cha69]), and Solomonoff ([Sol64]). Kolmogorov complexity measures the length of the shortest description of a string, measuring how much a given a finite sequence could be compressed. A string can be compressed significantly, if it is very redundant, if it has a lot of repitition. These are strings that intuitively we would not describe as "random". Thus Kolmogorv complexity seems to be a measure that captures the intuitive notion of randomness. A sequence such as 5555 5555 5555 5555 5555 does

not intuitively seem very random, while a sequence such as 1732 7356 2748 7621 6552 might appear more random, even though both are equally likely to be drawn from a uniform distribution. And Kolmogorov complexity would yield a very low measure for the first string and a fairly high one for the second string. As it turns out ([Mar66b]), The strings with high Kolmogorov complexity satisfy the statistical properties used to characterize statistically random sequences.

Kolmogorov complexity has matured into an established area of research with applications in various areas of computer science, and even the natural sciences in general. For an extensive treatment of the topic refer to the textbook by Li and Vintaí [LV97].

1.3 Contributions of this thesis

In this thesis we study resource-bounded notions of Kolmogorov complexity. There is a significant body of work in the literature addressing the notions of resource-bounded Kolmogorov complexity ([Sip83, Lev84, Ko86, Lon86, All89, All92, BM95, BT01, BFL02]). We consider variants of a measure Kt initially introduced by Levin [Lev84]. It has been shown before [All89] that there is a connection between the Kt complexity of the most simple strings of dense sets in $\mathsf{P}_{/\mathsf{poly}}$ and the existence of pseudo-random generators. In this thesis we will explore this connection further.

As mentioned above, many constructions of pseudo-random generators trade the computational hardness of a function for (pseudo-)randomness to make a probabilistic algorithm deterministic. While pseudo-randomness in the context of derandomization is formalized in a significantly different way than the notion of randomness we consider in the context of Kolmogorov complexity, in this thesis we establish a very close connection between the computational hardness of a function and the Kolmogorov complexity of its truth table. We consider the complexity measure KT, which is a variant of Levin's measure Kt. We show that for any finite Boolean function f, if χ_f represents the truth-table of f then the measure $\mathrm{KT}(\chi_f)$ is closely related (roughly

quadratic) to the circuit complexity of f. We establish similar relationships between other non-uniform complexity measures and other variants of resource-bounded Kolmogorov complexity measures.

Despite the fact that resource-bounded Kolmogorov complexity has been studied for over twenty years, it is still not quite as well understood as the traditional, resource-unbounded measure. It is assumed that the resource bounded variants, such as Kt for instance, behave in a very similar manner to the original measure K - in particular if one considers fairly generous resource bounds. However, in this thesis we show that this intuition is not always true. For the traditional measure K, it has been known since [ZL70] that a property called *symmetry of information* holds. Intuitively, this is the notion that for any two strings x and y, the string x carries about as much information about y as y carries about x. One way to formalize this idea is to state, that generating the string xy is not any easier than first generating x and then generating y: $\mathrm{K}(x) + \mathrm{K}(y|x) \leq \mathrm{K}(xy)$. Longpré showed ([Lon86]) that this property holds in some sense for Kolmogorov complexity measures with (generous) fixed resource bounds. However, in this thesis we show that this *does not* hold true for Levin's measure Kt and its variants (for generous resource bounds). Thus some resource-bounded Kolmogorov complexity measures differ significantly in certain aspects from the resource-unbounded notion.

Another point that shows that the measure K is better understood than the resource-bounded measures, is the question of how hard or easy it is to compute the measure for a given string. It is easy to argue that it is possible to decide in co-RE if for a given m the value $\mathrm{K}(x) > m$. Furthermore, it can be argued that this question can not be computed in RE. It was shown ([Mar66a, Kum96]) that the set of K-random strings is hard for co-RE. On the other hand, for the set of strings with high Kt complexity, no comparable results had been known. In most cases it is straightforward to establish the expected upper bound (for instance, $\mathrm{Kt}(x)$ can be

computed in exponential time by enumerating all descriptions in increasing length and determining which one generates x within the resource bound). However, it is not at all clear how to establish a meaningful lower bound. (For instance, at present we can only say that $\mathrm{Kt}(x)$ can not be computed by AC^0 circuits.) In this thesis we approach this question by showing that the problem of computing the resource-bounded complexity of a string is complete for different complexity classes.

For instance, we use the pseudo-random generators that were developed based on the construction presented by Nisan and Wigderson [NW94] to show that the set of strings with high Kt-complexity is complete for deterministic exponential time under reductions computable by polynomial size circuits and under reductions computable by nondeterministic polynomial time machines. This provides significant insight into the complexity of these types of sets. Intuitively, the sets containing strings with high Kolmogorov complexity have a very high information content - after all, each individual strings carries a lot of information - but due the seeming lack of internal structure of these sets, it is not obvious how to exploit this information through a (non-uniform) efficient reduction. Thus the completeness results that are obtained strongly contrast with the notion that typical complete sets exhibit a very clear combinatorial structure representative of the complexity class they reside in. Another point worth mentioning in this context is the fact that the set of random strings provides a natural example separating separating different completeness degrees for complexity classes. For instance we argue that R_{Kt} is not complete for EXP under truth-table reductions computable in time 2^{n^k}, even when provided with an advice string of length n^k. Yet, R_{Kt} *is* complete for EXP under truth-table reductions computable in nondeterministic polynomial time. To the best of our knowledge, the sets of strings with high Kolmogorov complexity seem to provide the first example of sets that were not explicitly constructed by diagonalization that provide a separation of different reductions.

Another aspect that we study is the question of what the complexity of the simple strings in a set implies about the complexity of certain complexity classes. We build on work of Allender in [All01], where it was shown that exponential time computation can be simulated by polynomial size circuits if and only if every P-printable set contains infinitely many simple strings[1]. In a similar spirit we provide results relating the complexity of search problems in exponential time to the complexity of simple strings of different subclasses (as opposed to P-printable) of sets in P.

The remainder of the thesis is structured as follows. Section 2 provides some general background. It introduces the conventions and notations used in the thesis as well as discussion of the utilized models of computation. Section 3 introduces the different notions of resource-unbounded and resource-bounded Kolmogorov complexity. In this section we also prove several theorems relating the different notions. The section concludes with a discussion of symmetry of information. In Section 4 we discuss the implications of a polynomial relation between certain K-complexity measures. We show that these relations are directly linked to the question about the complexity of the most simple strings in certain sets as well as inclusion properties of certain complexity classes. In Section 5 we study the sets $R_{\mathrm{K}\mu}$, the sets of strings with high complexity with respect to the measure $R_{\mathrm{K}\mu}$. We examine the computational complexity of these sets by proving completeness and non-completeness results for different complexity classes.

[1]Here "simple" means that $\mathrm{KT}(x) \leq (\log n)^{\mathcal{O}(1)}$

2 Definitions, Conventions and Notations

In this section we briefly discuss some notational conventions for this thesis. Unless stated explicitly otherwise, the definition of complexity classes, formalisms for computation, etc. are used as in the standard textbooks ([BDG95, Pap94, DK00]).

We will work with the standard alphabet $\Sigma = \{0, 1\}$. Languages or problems are subsets of Σ^*, λ denotes the empty string. For a language L, denote $L^{=n} = L \cap \Sigma^n$. Further, let $L^{\leq n} = \bigcup_{i \leq n} L^{=i}$. We will use the standard lexicographical ordering on strings. That is $x \leq_{\text{lex}} y$ if $|x| < |y|$ or $|x| = |y|$ and $x \leq y$ according to the standard phone-book order.

Given a set A, we will use $\chi_A(x)$ to denote the characteristic function of A that is defined as $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ if $x \notin A$. If $x_1, x_2, \ldots$ denotes all strings in Σ^* in lexicographic order, then $\chi_A = \chi_A(x_1)\chi_A(x_2)\ldots$ denotes the infinite characteristic sequence of A. Given a finite Boolean function f on n inputs, we use χ_f to denote the binary string of 2^n bits that represents the truth table of f.

Given a set A, the census function $\text{cens}_A(n)$ indicates how many strings of length n the set A contains - that is $\text{cens}_A(n) = |A^{=n}|$. The *density* of a set A is the function $\text{density}_A(n) = \frac{\text{cens}_A(n)}{2^n}$. A set A has *polynomial density*, if $\text{density}_A(n) = \frac{1}{n^{\mathcal{O}(1)}}$.

We will use $\text{int}(\cdot)$ to denote the interpretation of a string $w = w_1 \ldots w_n$ as $\text{int}(w) = 2^n - 1 + \sum_{i=1}^{n} 2^{n-i} \cdot w_i$. We will use $\text{bin}(\cdot)$ to denote an efficient encoding of integers as strings such that $x = \text{bin}(\text{int}(x))$ and $i = \text{int}(\text{bin}(i))$ for all $i \geq 0$.

When considering randomized computation we will use random strings that are drawn from the uniform distribution U_n. The random variable U_n assigns the constant probability $U_n(x) = \frac{1}{2^n}$ for all $x \in \{0, 1\}^n$. We will utilize the notation $\underset{r \in U_n}{\mathbf{Pr}} \left[P(r) \right]$ to indicate event $P(r)$ happens with the given probability when string r is drawn uniformly at random from $\{0, 1\}^n$.

Let $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \mapsto \Sigma^*$ denote a binary pairing function that is computable and

invertible in linear time and logarithmic space and for which $|\langle x_1, x_2\rangle| \leq |x_1| + |x_2| + \mathcal{O}(|x_1 x_2|)$. More precisely, we will use, $\langle x_1, x_2\rangle = 1^{|x_1|}0x_1x_2$. This function can recursively be extended for n-ary tuples as $\langle x_1, \ldots, x_n\rangle = 1^{x_1}0x_1 \langle x_2, \ldots, x_n\rangle$.
When we use the notation $M(x_1, \ldots, x_n)$ for a machine receiving several inputs $x_1, \ldots, x_n$, we consider the machine M receiving the string $\langle x_1, \ldots, x_n\rangle$.

2.1 Model of Computation

We formalize computation through a multi-tape Turing machine with two way infinite tapes that has random access to all its tapes. A similar machine model has been considered in [BIS88]. Formally, the cells on each tape are numbered relative to the initial position of the read-write head. For each tape the machine has an additional write-only address tape. When considering space usage, the used cells on the address tape are not being counted – this is consistent with the standard approach of providing space-bounded oracle machines with a write-only oracle tape and not counting the space used on that tape. In addition to the regular left/right moves, the machine has a *jump* move, where the read/write head of the tape is placed on the tape cell corresponding to the address on the address tape. In the remainder of the thesis, the term *Turing machine* always refers to a machine with random tape access.
Providing random access to the tapes of the machine enables meaningful computation in sub-linear time. We will consider resource-bounded Kolmogorov complexity measures, where the universal machine needs to quickly produce each individual bit of a string from one description for the entire string. Allowing random access to the tapes enables the machine to access possibly very few non-adjacent parts of the description very quickly when computing a bit of the string. Most results in this thesis also hold for standard Turing machines with sequential access to the tapes. Results that involve explicit simulation overheads might need to be slightly restated when considering standard machines.

We consider oracle Turing machines as the standard extension of the Turing machine model. In the event that we consider a machine that has access to two oracles, we consider $M^{(A,B)}$ to denote the machine $M^{A \bowtie B}$ that has access to the oracle $A \bowtie B = \{0u \mid u \in A\} \cup \{1u \mid u \in B\}$.

While it is possible to study the length of the shortest description of a string that can be decoded by an arbitrary given Turing machine, it is much more meaningful to consider such a measure with regard to a universal machine. This will ensure that there is single decoding machine machine, that can interpret any encoding of any description with very small overhead. Thus the optimal (within small additive terms) description for a string does not depend the actual machine used for decoding the description.

Definition 2.1 (Universal Turing Machine). *A Turing machine U is universal, if for every Turing machine M there is a finite encoding d_M, such that $U(d_M, x) = M(x)$ for every x. We will assume without loss of generality, that the input to U can be padded. That is, we assume that for every description $d = \langle d_M, x\rangle$ the machine U behaves identically when given a padded input: $U(0^k 1d) = U(d)$ for any k.*

The following fact provides a similar simulation overhead as the result for sequential Turing machines ([HS66]), but the proof is much simpler.

Fact 2.2 (Minimal Simulation Overhead). *Let A be any oracle. There is a universal (random access) Turing machine U with 3 work tapes, such that if M^A is a (random access) Turing machine running in time t and space s on input x, then U^A simulates M^A in $\mathcal{O}(t \log t)$ steps and $\mathcal{O}(s)$ space.*

If M^A is a (random access) Turing machine with at most two tapes running in time t and space s, then U^A simulates M^A in $\mathcal{O}(t)$ steps and $\mathcal{O}(s)$ space.

Proof. The machine U has 3 worktapes T_M, T_{work} and T_{aux}. On input $d = \langle M, x\rangle$, the machine U copies the description of M on tape T_M and determines k, the number

of tapes of M. If $k = 2$, then the machine uses the two remaining tapes T_{work} and T_{aux} to directly simulate M in a straightforward manner. If $k > 2$ the universal machine proceeds as follows. It views the main work tape T_{work}, as k interspersed virtual tapes. The machine U then sets up k counters on the auxiliary tape, to keep track of the position of the k virtual tape-heads on T_{work}. Finally, U copies x on the virtual tape corresponding to the input tape of M. It then starts the step-by-step simulation by reading the data from each of the virtual heads, determining the next step and the updating of each virtual tape. Since the original machine M runs in space s, the address of each virtual head is at most $\log s$ bits long. Thus reading the virtual heads and updating the virtual tapes can be done in $\mathcal{O}(\log s)$ time. Therefore the overall simulation requires $\mathcal{O}(t \log s) \leq \mathcal{O}(t \log t)$ time. There is no significant space overhead, and thus the simulation requires at most $\mathcal{O}(s)$ space. This argument also holds relative to any oracle A. □

In most settings in this thesis we model non-uniform computation with Boolean circuits. We use the standard definitions of unbounded fan-in circuits. In some applications these circuits have oracle gates. For a good overview of circuit complexity refer to [Vol99]. We consider the size of a circuit to refer to the number of wires in the circuit.

Definition 2.3. *For a function f (or the string χ_f representing its truth-table), define* $\mathrm{SIZE}^A(f)$ *to be size of the smallest circuit with oracle gates for A that computes f. We use* $\mathrm{SIZE}(f)$ *to denote* $\mathrm{SIZE}^\emptyset$.

Another non-uniform measure of computation that better relates to small space-bounded computation is the notion of branching programs. A branching program for n inputs is a binary, directed, acyclic graph. Each inner node has out-degree 2 and is marked with one of the n variables. The two arcs leaving such an inner node are labeled 0 and 1. The graph has a designated root node. Two leaves are marked

as accepting and rejecting respectively. The branching program accepts an input $x = x_1 \dots x_n$, if the path that starts at the root node and an each node labeled x_i follows the arc that corresponds to the value of the variable x_i reaches the accepting node. The size of a branching program is the number of nodes. For more background on branching programs, please refer to [DK00]

We will need some facts that relate how quickly a Turing machine can simulate a circuit and vice versa.

Fact 2.4 (Simulation of Circuits). *Let A be any oracle. If C^A is a circuit of size m computing a function of n input bits, then there is an encoding d_C of C of size $\mathcal{O}(m(\log m + \log n))$, and a two tape (random access) Turing machine M, such that $M^A(d_C, x)$ outputs $C^A(x)$ in time $\mathcal{O}(m(\log m + \log n))$ for $|x| = n$ and outputs $*$ for $|x| \neq n$.*

Proof. Let C be an oracle circuit of size m computing a function of n input bits. Let $g_1, \dots, g_m$ be an ordered sequence of gates, where $g_1, \dots, g_n$ are the input gates, g_m is the output gate, and C contains a wire (g_j, g_k), only if $j < k$.

We can describe the entire circuit by listing for each gate g_k all wires that feed into it. A wire (g_j, g_k) can be encoded as a quadruple (k, t, i, j), where k is the index of that gate g_k, t denotes the type of g_k (AND, OR, NOT, INPUT, ORACLE), i denotes that the wire provides the i-th input bit to g_k, and j is the index of that gate g_j. This description requires at most $\mathcal{O}(\log m + \log n)$ bits. As the circuit has m gates, the description d_C requires $\mathcal{O}(m(\log m + \log n))$ bits.

In order to evaluate the circuit, the machine evaluates the gates in order $g_1, \dots, g_m$. To evaluate a gate, we evaluate each wire feeding into the gate. Since the machine has random access to its tapes, the evaluation of one wire can be done in linear time in terms of its description length, i.e. in time $\mathcal{O}(\log m + \log n)$. In order to evaluate an ORACLE-gate, we write the values of wires feeding into the gate onto the query-tape

of the machine to ask the oracle. Note that the algorithm can be easily implemented using two tapes. The overall runtime of the algorithm is $\mathcal{O}(m(\log m + \log n))$. □

Fact 2.5 (Simulation by Circuits). *Let A be an oracle. If M^A is random-access oracle Turing machine running in time $t(n)$ on inputs of size n, then for every n there is a circuit C_n with oracle A of size $t(n)^2 \log n$, computing M^A.*

Proof. Let M be a k-tape random-access oracle Turing machine running in time $t(n)$. Then there is a k-tape sequential machine simulating M in time $t(n)^2$ using space $\mathcal{O}(t(n))$. Using the main result in [PF79], this machine can be simulated by a 2 tape oblivious oracle machine M' in time $\mathcal{O}(t(n)^2 \log t(n))$, still in space $\mathcal{O}(t(n))$. (Note that the construction in [PF79] relativizes.) This gives rise to the standard construction of a circuit that simulates the computation tableaux of the machine. The circuit is constructed in levels where level i represents the machine during step i. The first level consists of gates representing the entire initial configuration of size $\mathcal{O}(t(n))$. As M' is oblivious, the position of the read write heads is known in advance for each step i. Each level i just needs a subcircuit of size $\mathcal{O}(1)$ to compute the changes in the configuration of M. Thus the entire circuit contains $\mathcal{O}(t(n)^2 \log t(n))$ gates. Except for the ORACLE-gates, each gate has only constant fan-in. Thus the number of wires feeding into these gates is also $\mathcal{O}(t(n)^2 \log t(n))$. The initial machine made at most $t(n)$ queries to the oracle - each of size at most $t(n)$. Thus the final circuit contains at most $t(n)$ oracle gates with at most $t(n)$ input bits. Thus the oracle gates require a total of at most $t(n)^2$ many wires. Therefore the size of the final circuit is $\mathcal{O}(t(n)^2 \log t(n))$. □

Note that the simulation overheads provided in Fact 2.4 and Fact 2.5 are explicitly stated for random-access Turing machines. For the sequential machine model the upper bound for the circuit size in Fact 2.4 would drop to $\mathcal{O}(t \log t)$ whereas the time required to simulate a circuit in Fact 2.5 would rise to $\mathcal{O}(t^2 \log t)$.

2.2 Complexity Classes and Reductions

We will use the standard definitions for complexity classes. as mentioned below. For a more detailed coverage of these notions, the reader is referred to the standard textbooks [BDG95, Pap94, DK00].

Definition 2.6. *The class* $\mathsf{DTime}[t(n)]$ *denotes the class of all sets that can be decided by a deterministic Turing machine in* $\mathcal{O}(t(n))$ *steps. The class* $\mathsf{NTime}[t(n)]$ *denotes the class of all sets that can be decided by a nondeterministic Turing machine in* $\mathcal{O}(t(n))$ *steps. The class* $\mathsf{DSpace}[s(n)]$ *denotes the class of all sets that can be decided by a deterministic Turing machine using* $\mathcal{O}(s(n))$ *space.*

This gives rise to the following definitions.

Definition 2.7 (Deterministic and Nondeterministic Complexity Classes).

$$\mathsf{P} = \bigcup_{c \in \mathbb{N}} \mathsf{DTime}[n^c] \qquad \mathsf{NP} = \bigcup_{c \in \mathbb{N}} \mathsf{NTime}[n^c]$$

$$\mathsf{PSPACE} = \bigcup_{c \in \mathbb{N}} \mathsf{DSpace}[n^c]$$

$$\mathsf{E} = \bigcup_{c \in \mathbb{N}} \mathsf{DTime}[2^{cn}] \qquad \mathsf{NE} = \bigcup_{c \in \mathbb{N}} \mathsf{NTime}[2^{cn}]$$

$$\mathsf{EXP} = \bigcup_{c \in \mathbb{N}} \mathsf{DTime}[2^{n^c}] \qquad \mathsf{NEXP} = \bigcup_{c \in \mathbb{N}} \mathsf{NTime}[2^{n^c}]$$

We will consider the following probabilistic complexity classes.

Definition 2.8 (Probabilistic Complexity Classes). *Let* C *be any class of* $\{\mathsf{BPP}, \mathsf{RP}, \mathsf{ZPP}, \mathsf{MA}\}$. *A set* A *belongs to* C*, if there is a deterministic machine* M *and a constant* c*, such that* M *runs in time* n^c *in terms of the input and the following holds:*

$\mathsf{C} = \mathsf{BPP}$: $x \in A \implies \underset{r \in U_{|x|^c}}{\mathbf{Pr}} \big[M(x,r) \text{ accepts} \big] \geq \frac{2}{3}$
$x \notin A \implies \underset{r \in U_{|x|^c}}{\mathbf{Pr}} \big[M(x,r) \text{ rejects} \big] \geq \frac{2}{3}.$

$\mathsf{C} = \mathsf{RP}$: $x \in A \implies \underset{r \in U_{|x|^c}}{\mathbf{Pr}} \big[M(x,r) \text{ accepts} \big] \geq \frac{2}{3}$
$x \notin A \implies \underset{r \in U_{|x|^c}}{\mathbf{Pr}} \big[M(x,r) \text{ rejects} \big] = 1$

$\mathsf{C} = \mathsf{ZPP}$: $L \in \mathsf{RP}$ *and* $\overline{L} \in \mathsf{RP}$.

Note that this implies, that there is a probabilistic machine M that decides M with an error-probability of 0 *and an expected run-time that is polynomial in the input.*

$\mathsf{C} = \mathsf{MA}$: $x \in A \implies \exists w \in \{0,1\}^{|x|^c} \underset{r \in U_{|x|^c}}{\mathbf{Pr}} \big[M(x,r,w) \text{ accepts} \big] \geq \frac{2}{3}$
$x \notin A \implies \forall w \in \{0,1\}^{|x|^c} \underset{r \in U_{|x|^c}}{\mathbf{Pr}} \big[M(x,r,w) \text{ rejects} \big] \geq \frac{2}{3}.$

We also define reductions in the standard manner.

Definition 2.9 (Standard Reductions). *Let A and B be sets, and* r *be a class of resource bounds.*

(i) A is many-one reducible to B ($A \leq_{\mathrm{m}}^{\mathsf{r}} B$), If there is a function f computable within resource bounds r*, such that $x \in A \iff f(x) \in B$.*

(ii) A is Turing reducible to B ($A \leq_{\mathrm{T}}^{\mathsf{r}} B$), If there is a Turing machine with oracle B that decides A within resource bounds r*.*

(iii) A is truth-table reducible to B ($A \leq_{\mathrm{tt}}^{\mathsf{r}} B$), If there is a Turing machine with oracle B that decides A within resource bounds r *and every query q_i can be generated independently of the answer to previous queries.*

In certain applications we will consider sets that can be reduced to themselves in a nontrivial way. We will formalize these properties with the following notions:

Definition 2.10 (Self-reducibility). *Let A be a set.*

(i) *A is polynomial time self-reducible, if $A \leq_{\mathrm{T}}^{\mathsf{P}} A$ and on input x the reducing machine M asks only queries $q_i \neq x$.*

(ii) *A is polynomial time downward self-reducible, if $A \leq_{\mathrm{T}}^{\mathsf{P}} A$ and on input x the reducing machine M asks only queries q_i with $q_i \leq_{\mathrm{lex}} x$.*

(iii) *A is polynomial time random self-reducible, if there is a constant c and a probabilistic Turing machine M running in polynomial time, such that $\mathbf{Pr}\big[\, M^B(x)$ accepts iff $x \in A \,\big] \geq \frac{2}{3}$ for every oracle B that agrees with A on most strings. The set B agrees with A on most strings, if $|(A^{=n} \oplus B^{=n})| \leq \frac{2^n}{n^c}$ where "$\oplus$" denotes the symmetric difference of A and B.*

2.3 Resource Bounded Measure

In this section we describe the fragment of Lutz's measure theory that we will need. For a more detailed presentation of this theory we refer the reader to the survey by Lutz [Lut97]. The measure on resource-bounded complexity classes is obtained by imposing appropriate resource-bound on a game theoretical characterization of the classical Lebesgue measure.

Definition 2.11 (Martingale). *A martingale is a function $d : \{0,1\}^* \to \mathbb{Q}$ such that for every $w \in \{0,1\}^*$*

$$d(w) = \frac{d(w0) + d(w1)}{2} \quad .$$

We say that a martingale d succeeds on a language A, if

$$\limsup_{w \sqsubseteq \chi_A, w \to \chi_A} d(w) = \infty \quad .$$

(Note that $w \sqsubseteq A$ denotes that w is a prefix of χ_A).

This definition can be motivated by the following betting game in which a gambler puts bets on the successive membership bits of a hidden language A. The game proceeds in infinitely many rounds where at the end of round n, it is revealed to the gambler whether $s_n \in A$ or not. The game starts with capital 1. Then, in round n, depending on the first $n-1$ outcomes $w = \chi_A[0 \dots n-1]$, the gambler bets a certain fraction $\varepsilon_w d(w)$ of his current capital $d(w)$, that the nth word $s_n \in A$, and bets the remaining capital $(1-\varepsilon_w)d(w)$ on the complementary event $s_n \notin A$. The game is fair, i.e. the amount put on the correct event is doubled, the one put on the wrong guess is lost. The value of $d(w)$, where $w = \chi_A[0 \dots n]$ equals the capital of the gambler after round n on language A. The player wins on a language A if he manages to make his capital arbitrarily large during the game.

Definition 2.12. *A class* C *has measure* 0 *in* EXP *(*$\mu(\mathsf{C}|\mathsf{EXP})$*) if there is a martingale* d *such that* $d(w)$ *is computable in time* $2^{(\log|w|)^{\mathcal{O}(1)}}$ *and such that* d *succeeds on every set* $A \in \mathsf{C}$.

We need the following lemma due to Lutz [Lut87] that states that an "easy" infinite union of measure 0 sets also has measure 0.

Lemma 2.13. *Let* $(d_i)_{i\in\mathbb{N}}$ *be a collection of martingales and* $\mathsf{C} = \bigcup_{i\in\mathbb{N}} \mathsf{C}_i$. *If each* d_i *succeeds on* C_i *and* $d(i,w) = d_i(w)$ *can be computed in time* $2^{(\log|w|)^{\mathcal{O}(1)}}$, *then* $\mu(\mathsf{C}|\mathsf{EXP}) = 0$.

3 Resource Bounded Kolmogorov Complexity

The basic idea of Kolmogorov complexity (sometimes referred to as K-complexity) is to measure the complexity of individual strings by how hard each string is to describe. That is, strings that can be generated by a very simple algorithm have low K-complexity, whereas strings that require a very detailed description have high K-complexity. Formally, this measure $\mathrm{K}(x)$ is defined as the length of the shortest description d_x, from which a fixed universal Turing machine U can generate x. While motivated from different perspectives, this approach was introduced independently by Solomonoff [Sol64], Kolmogorov [Kol68], and Chaitin [Cha66, Cha69]. Since then, Kolmogorov complexity has proven itself to be a very useful tool in various areas of theoretical computer science as well as a fruitful research area in its own right. For an in-depth treatment of Kolmogorov complexity refer to [LV97].

From the perspective of computational complexity theory, there is one drawback to the classic notion of Kolmogorov complexity. The measure $\mathrm{K}(x)$ is defined in terms of recursion theory. It does not give any consideration to the resources that are required to generate x from its description d_x. As long as x is computable by the universal machine U from d_x, it does not matter how long this computation might take. To remedy this, There have been several approaches of considering resource-bounded notions of Kolmogorov complexity in order to define a measure in the context of complexity theory. In this section we will first formally introduce the classical notion of K-complexity. Then we will discuss different resource-bounded versions.

3.1 Properties of Kolmogorov Complexity

Let us formally define the classical notion of Kolmogorov complexity and consider some of its properties.

Definition 3.1 (Kolmogorov Complexity). *Let U be a Turing machine. Define the* Kolmogorov complexity of x relative to U *as*

$$\mathrm{K}_U(x) = \min\{|d| \mid U(d) = x\} \quad .$$

If U is a universal machine (→Definition 2.1), the specific choice of U just changes the complexity measure by an additive constant:

Fact 3.2. *Let U and U' be two Turing machines. If U is a universal machine, then there exists a constant c, such that $\mathrm{K}_U(x) \leq \mathrm{K}_{U'}(x) + c$ for all x.*

As we consider the complexity of strings asymptotically, the additive constant does not carry significant weight. Thus, the specific choice of the universal machine is not relevant when considering the measure $\mathrm{K}_U(x)$. We follow convention (as in [LV97]) and drop the subscript U.

However, there are situations considered in the literature where the particular choice of U actually does matter. For instance, in [ABK03] it was shown that for every computable time bound t, there exists a universal machine U and a decidable set A, such that A is not *dtt*-reducible to R_{K_U} in time t, even though R_{K_U} is hard under *dtt*-reductions for co-RE. However, all results in this thesis hold for any choice of a universal Turing machine.

In some contexts it will be of interest to measure the complexity of a string relative to a given string. One might be interested in determining whether two strings carry roughly the same information (as opposed to the same amount of information). This can be captured with the notion of *conditional Kolmogorov complexity*.

Definition 3.3 (Conditional Kolmogorov Complexity). *Let U be a Turing machine and A be an arbitrary oracle. Define the* Kolmogorov complexity of x given y

relative to A as

$$\mathrm{K}_U^A(x|y) = \min\{|d| \mid U^A(d,y) = x\} \quad .$$

Again, we fix a universal machine U and write $\mathrm{K}^A(x|y)$ instead of $\mathrm{K}_U^A(x|y)$. If $A = \emptyset$, we abbreviate $\mathrm{K}^A(x|y)$ as $\mathrm{K}(x|y)$.

There are some very basic properties of Kolmogorov complexity that can be easily pointed out. For instance, there is a trivial upper bound of the K-complexity of any string, since every string can be described by itself.

Fact 3.4. *There is a constant c, such that $\mathrm{K}(x) \leq |x| + c$ for every string x.*

A simple counting argument shows, that almost every string has high Kolmogorov complexity.

Fact 3.5. *For every m and every n, there are at least $(2^{n+1} - 2^{m+1})$ strings x of length $|x| \leq n$, with $\mathrm{K}(x) > m$.*

3.2 Resource Bounded K-complexity

Several different notions of resource-bounded Kolmogorov complexity have been studied in the literature. While there are many different aspects that can vary in the definition of a resource-bounded Kolmogorov complexity measure (What is the machine model used? Does the universal machine need to generate the string, or merely recognize it?), we will focus here on how time restriction can be incorporated into the classical measure.

One approach was introduced in [Ko86]. There, a fixed time bound was imposed on the runtime of the Universal machine, i.e. the measure $\mathrm{K}_{\mathrm{t}}^{t(n)}(x)$ is the length of the shortest description of x that can be decoded in time $t(|x|)$. This measure thus does not distinguish if the description can be decoded very quickly (i.e. in

time $|x|$) or if it requires maximal time (i.e. $t(|x|)$). Furthermore the choice of the time bound considered is fairly arbitrary and slight changes might dramatically change the measure for a particular string. Typically this measure is considered for a class of functions (e.g., $\mathrm{K}_{\mathrm{t}}^{\mathsf{poly}}(x)$)). However this can make comparisons about the relative complexity of two different strings a little bit awkward, i.e. some results have to be stated as: "for every polynomial p there is a polynomial q, such that $\mathrm{K}_{\mathrm{t}}^{q}(y) \leq \mathrm{K}_{\mathrm{t}}^{p}(x)$". (For instance, the notion of symmetry of information is expressed in [LW95] as $\forall q \exists p\, \mathrm{K}_{\mathrm{t}}^{p}(x) + \mathrm{K}_{\mathrm{t}}^{p}(y|x) \leq \mathrm{K}_{\mathrm{t}}^{q}(xy) + \mathcal{O}(\log|xy|)$.

Sipser also considered a fixed time bounded complexity measure in [Sip83]. Instead of focusing on the information required to *produce* a string his measure $\mathrm{KD}_{\mathrm{t}}^{t(n)}(x)$ considered the length of the shortest description, such that x can be *distinguished* by the universal machine from every other string in time $t(|x|)$.

Hartmanis considered resource-bounded K-complexity from a different perspective. He introduced the notion $\mathrm{K}[d(n), t(n)]$ in [Har83], as the set of strings x that have descriptions of length $d(|x|)$ that are decodable in time $t(|x|)$. This approach has the advantage that the length of the description and the resource bound can be considered independently with very fine detail. However, it does not provide an explicit measure to determine the complexity of an individual string x.

The notion of K-complexity that we will use in this thesis was first introduced by Levin in [Lev84]. It provides an explicit measure of resource-bounded Kolmogorov complexity without the need to fix a parameter or a resource bound. Thus it does not have the drawbacks of the approaches mentioned above. Additionally, as we will show in this thesis, Levin's measure Kt has a close connection to the non-uniform complexity of a string. If we interpret x to be the truth-table of a function f_x, then the value $\mathrm{Kt}(x)$ gives an estimate of the size of the smallest oracle circuit C relative to E, computing f_x.

3.3 Levin's Kt-measure

One perspective on the descriptive complexity of a string could be the following: given an enumeration of all strings from simple to hard, where does the string appear in the enumeration? This notion can be formally captured by the *age* of a string. The age of a string x can be defined as $age(x) = \min\{2^{|d|} \cdot t \mid U(d) = x \text{ in } t \text{ steps}\}$. An algorithm that runs $U(d)$ for 2^i steps for each description of length at most i, will enumerate all strings by increasing age. Levin's notion of resource-bounded Kolmogorov complexity, considers the logarithm of the age of a strings and thus provides a measure that fits within the parameters of traditional Kolmogorov complexity.
Levin initially defined $\mathrm{Kt}_U(x) = \min\{|d| + \log t \mid U(d) = x \text{ in } t \text{ steps}\}$. In this thesis we consider other complexity measures that are inspired by this definition, but that require some additional technical details. In order to work with a consistent framework, we will define the measure Kt slightly different for the purposes of this thesis, but we remark that it differs from the original definition by at most an additive $\mathcal{O}(\log|x|)$.

Definition 3.6 (Kt)**.** *Let U be a Turing machine and let A be an oracle. Define the measure $\mathrm{Kt}^A_U(x|y)$ of a string x as*

$$\mathrm{Kt}^A_U(x|y) = \min\{|d| + \log t \mid \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } t \text{ steps, iff } x_i = b\}$$

We denote $x_{|x|+1} = *$. We use $\mathrm{Kt}_U(x|y)$ to denote $\mathrm{Kt}^{\emptyset}_U(x|y)$, and we use $\mathrm{Kt}_U(x)$ to denote $\mathrm{Kt}_U(x|\lambda)$.

As is the case with the resource-unbounded Kolmogorov complexity, the measure Kt_U is essentially invariant under the particular (reasonable) choice of U. There are universal Turing machines, that can simulate any other Turing machine M in time $\mathcal{O}(t \log t)$ (→Fact 2.2), where t is the original time required for the computation of M. This motivates the following definition.

Definition 3.7 (Kt**-optimal).** *A universal machine U is* Kt-optimal*, if for every Turing machine U' there is a constant c for which the following holds.*

$$\mathrm{Kt}_U(x) \leq \mathrm{Kt}_{U'}(x) + c\log|x| \quad .$$

We will fix a Kt*-optimal Turing machine U and use $\mathrm{Kt}(x)$ to denote $\mathrm{Kt}_U(x)$.*

Note that there is a trivial upper bound on the complexity of a string x.

Fact 3.8. *There is a constant c, such that for every x the following holds:*

$$\mathrm{Kt}(x) \leq |x| + c\log|x| \quad .$$

Levin's definition of Kt was motivated by the question of an optimal search strategy for NP search problems. Problems in NP can be formulated as questions about the existence of a witness w that satisfies a polynomial time computable predicate P for a given instance x, i.e. a w such that $P(x, w)$ is true. For instance, the problem Sat asks for the existence of a satisfying assignment to a propositional logic formula. Or the problem Clique asks if a given graph contains a clique of a given size. An NP-search problem, is the problem of finding a witness for an instance of a problem in NP. It is still not clear how to find such a witness efficiently. However, Levin presented a generic algorithm that searches for solutions to an arbitrary NP-search problem. The runtime of this algorithm is essentially optimal, i.e. it runs in time nearly $\mathcal{O}(t)$, if t is the runtime of the optimal algorithm.

Given x, Levin's algorithm essentially just enumerates all possible witnesses w in order of increasing $\mathrm{Kt}(x|w)$. For each such w it checks $P(x, w)$ before proceeding to the next possible witness. Assume that an optimal algorithm finding a witness for a predicate P requires time $t(|x|)$. Assume further that $P(x, w)$ can also be evaluated in time $t(|x|)$. Note that $\mathrm{Kt}(w|x) \leq c + \log t(|x|) = m_w$. Levin's algorithm checks w after testing at most all other descriptions w' with $\mathrm{Kt}(w'|x) \leq m_w$. Thus, the runtime

of it can be bounded above by $2^{m_w} \cdot 2^{m_w} \cdot t(|x|) = 2^{2(\log t(|x|)+1)} \cdot t(|x|) = \mathcal{O}(t(|x|)^3)$ Using a slightly more refined approach with different model of computation, it is possible to achieve a linear upper bound [LV97].

Levin's Kolmogorov measure Kt turns out to be a useful tool in relating different assumptions in various areas of complexity theory. We briefly mention two results due to Allender. There is a connection between pseudo random generators (PRGs) and Kt-complexity. Pseudo random generators are algorithms that produce output that seems random to a probabilistic algorithm. It is secure if no algorithm within certain resource bounds can distinguish the output of the generator from truly random bits. (For more background on pseudo random generators refer to Subsection 5.2.) In [All89] the existence of pseudo random generators is related to to existence of Kt-simple strings in dense sets from $\mathsf{P}/_{\mathsf{poly}}$.

Theorem 3.9 ([All89]). *If secure*[2] *pseudorandom generators exits, then every dense*[3] *set* $L \in \mathsf{P}/_{\mathsf{poly}}$ *contains a simple*[4] *string of every length* n *for which* $L^{=n} \neq \emptyset$.

Even though there are not any unconditional constructions for pseudo random generators (PRG) known, there are many conditional constructions ([NW94, BFNW93, IW97, IW98, SU01]) and it is widely believed PRGs exist unconditionally . Hence, it seems that every dense set in $\mathsf{P}/_{\mathsf{poly}}$ contains many simple strings.

On the other hand, there is a connection between Kt-complexity and the hardness of NE-search problems. NE-search problems can be characterized as problems, where for a given x we need to find a witness w of length $2^{\mathcal{O}(|x|)}$ such that a predicate $P(x, w)$, computable in time $2^{\mathcal{O}(|x|)}$, is satisfied. In [All89] it is shown that that the ability to deterministically find witnesses to NE-search problems in exponential time is equivalent to the existence of Kt-simple strings in sets from P.

[2]Here a pseudo random generator is secure, if every circuit of size at most $2^{\varepsilon n}$ has roughly the same acceptance probability for pseudo random strings and truly random strings.

[3]Here a set L is dense, if there is an ε, such that $|L^{=n}| > \frac{2^n}{n^{\mathcal{O}(1)}}$.

[4]Here a string x is simple if $\mathrm{Kt}(x) \leq \mathcal{O}(\log n)$.

Theorem 3.10 ([All89]). *It is possible to find witnesses for a* NE*-search problem in linear exponential time, if and only if every set* $L \in \mathsf{P}$ *contains a simple*[5] *string for every length* n *for which* $L^{=n} \neq \emptyset$.

There is some strong evidence that NEXP-search problems cannot be solved in deterministic exponential time. In particular there is an oracle relative to which NEXP-search problems require doubly-exponential time ([BGS75]). Thus it seems likely that there are sets in P that *can* avoid simple strings. By Theorem 3.9, it is unlikely that these sets are dense. However, this is sharply contrasted by the intuition that strings in sparse sets are fairly easy to describe by their index in the set. In fact, every string belonging to a sparse, computable set has low K-complexity. This motivates the study of Kt in an effort to understand these discrepancies. In particular, the question of the minimal complexity of strings in sets is covered in Section 4 in more detail.

How does the measure Kt relate to other resource-bounded complexity measures? In particular, how is Kt related to $\mathrm{K}_{\mathrm{t}}^{r(n)}$ for different resource bounds $r(n)$? It seems that the two measures are somewhat orthogonal. It is possible for any sufficiently large n to construct an $x \in \{0,1\}^n$ (by diagonalization), such that $\mathrm{K}_{\mathrm{t}}^{r(n)}(x) < \mathrm{Kt}(x)$. On the other hand, it might also likely be that there is a string $y \in \{0,1\}^n$, such that $\mathrm{Kt}(y) < \mathrm{K}_{\mathrm{t}}^{r(n)}(y)$. (However, a directly proof by diagonalization does not go through.) It is possible, though, to establish the following inequalities relating Kt and $\mathrm{K}_{\mathrm{t}}^{r(n)}$.

Fact 3.11.

(i) *If* $\mathrm{K}_{\mathrm{t}}^{r(n)}(x) \leq m$, *then* $\mathrm{Kt}(x) \leq m + \log r(n)$.

(ii) *If* $\mathrm{Kt}(x) \leq b(|x|)$ *for some function* b, *then* $\mathrm{K}_{\mathrm{t}}^{2^{b(n)}}(x) \leq b(|x|)$.

(iii) $\mathrm{K}_{\mathrm{t}}^{r(n)}(x) \leq \mathrm{Kt}(x)$ *for any* $r(n) \geq n^c \cdot 2^n$.

[5] Again, here a string x is simple if $\mathrm{Kt}(x) \leq \mathcal{O}(\log n)$.

Proof. Item (i) and (ii) follow directly from the definition of the measures. Item (iii) follows from the fact that $\mathrm{Kt}(x) \leq |x| + c\log|x|$ for every x. Thus the time required to decode the optimal description d_x for x is bounded from above by $2^{|x|+c\log|x|}$. It can therefore be decoded in time $r(n)$ as long as $r(n) \geq n^c \cdot 2^n$. □

3.4 KT and other variations of Kt

The resource-bounded Kolmogorov complexity measure Kt measures the length of the shortest description that is easy to decode. In order to measure the complexity of a given string, one could also take the following approach, which at first glance seems somewhat different than Levin's idea.

Given a string x, view it as the truth-table of a Boolean function f_x with $\log|x|$ variables. (For simplicity assume $|x| = 2^k$.) One could consider the circuit complexity $\mathrm{SIZE}(f_x)$ to measure the complexity of x. As circuits can be evaluated in polynomial time, viewing circuit complexity as a resource-bounded Kolmogorov complexity measure, puts a fairly restrictive resource bound on the measure.

Since circuit complexity has been intensively studied, this approach seems to provide a fairly refined tool for the study of Kolmogorov complexity. It would thus be desirable to place this measure more naturally into the existing framework. As it turns out, one can define a complexity measure KT in the spirit of Levin's Kt measure, that essentially captures the circuit complexity of a string.

Definition 3.12 (KT)**.** *Let U be a Turing machine and A an oracle. Define the measure $\mathrm{KT}^A_U(x|y)$ of a string as*

$$\mathrm{KT}^A_U(x|y) = \min\big\{|d| + t \;\big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } t \text{ steps, iff } x_i = b\big\}$$

Again, we denote $x_{|x|+1} = *$. We omit the superscript A, if $A = \emptyset$ and the string y, if $y = \lambda$.

The definition of KT requires the universal machine to generate x bit by bit in time t. This technicality is necessary to make the measure meaningful. If U was required to output the entire string x, it would require a runtime of at least $|x|$. However, any string x can be generated from a description $d = x$ in time $|x|$. Thus the measure would be very limited in its range.

The capital "T" in the notation for KT reflects the fact that the time component weighs more strongly than in the Kt measure.

As is the case with the measure Kt, the measure KT_U is essentially invariant under the particular (reasonable) choice of U. In this case however, the time overhead for simulating Turing machines has greater impact. This larger overhead has an impact on the upper bound that can be derived for the complexity of a string when it is used to describe itself. However, as it is desirable to have a uniform upper bound on the complexity of a string over different measures, we will add this as a requirement for the optimal machine.

Definition 3.13 (KT**-optimal).** *A universal machine U is* KT-optimal*, if two criteria hold. First, for every Turing machine U' there is a constant c for which the following holds.*

$$\mathrm{KT}_U(x) \leq c \cdot \mathrm{KT}_{U'}(x) \log \mathrm{KT}_{U'}(x) \quad .$$

Second, for a KT*-optimal machine we require $\mathrm{KT}_U(x) \leq |x| + c' \log n$ for every x and a universal constant c'.*

We will fix an arbitrary KT-optimal Turing machine U and denote $\mathrm{KT}(x) = \mathrm{KT}_U(x)$.

Remark 3.14. We will consider KT relative to different oracles A. For some of these oracles, it turns out that there is a machine U (depending on A), s.t. for every machine U' we have $\mathrm{KT}^A_U(x) \leq c \cdot \mathrm{KT}^A_{U'}(x)$. In these cases we will consider KT^A relative to such an optimal machine U.

If one views a given string x as the truth table of a function f_x, then $\mathrm{KT}(x)$ roughly corresponds to circuit complexity of f_x.

Theorem 3.15. *For any given w, define f_w as the Boolean function $f_w(\mathrm{bin}(i)) = w_i$ for $1 \leq i \leq |w|$ and $f_w(i) = 0$ for $|w| < i \leq 2^{\lceil \log |w| \rceil}$. Let A be an oracle and let x and y be given, with f_x and f_y being the corresponding Boolean functions. Let $n = |x|$.*

(i) $\mathrm{SIZE}^{(A,f_y)}(f_x) \leq \mathcal{O}((\mathrm{KT}^A(x|y))^2 \log \mathrm{KT}^A(x|y))$,

(ii) $\mathrm{KT}^A(x|y) \leq \mathcal{O}(\mathrm{SIZE}^{(A,f_y)}(f_x) \log(\mathrm{SIZE}^{(A,f_y)}(f_x)) + \log n)$.

Proof.

(i) Assume a given string x has $\mathrm{KT}^A(x|y) = m$. Thus there is a description d_x, such that $U^A(d_x, i, y, b)$ accepts iff $b = x_i$ in at most m steps. By Fact 2.5 the computation of $U^{(A,f_y)}$ can be implemented in a circuit of size $\mathcal{O}(m^2 \log m)$ that already contains the information of d_x hard-wired. Thus $\mathrm{SIZE}^{(A,f_y)}(f_x) = \mathcal{O}(m^2 \log m)$.

(ii) Assume that there is a circuit C_x of size m with oracle gates for A and f_y, such that on input i (in binary) the circuit computes $C(i) = x_i$ for $i \leq |x|$. Denote $n = |x|$.

By Fact 2.4 there is an encoding d of C of size $\mathcal{O}(m \log m)$ and a two-tape Turing machine M that, given oracle access to A and f_y, accepts $\langle d, i \rangle$ if and only if $C(i) = 1$. Note that M runs in time $\mathcal{O}(m \log m)$.

Let $d' = \langle d, \mathrm{bin}(n) \rangle$. Consider the machine M' that given oracle A and f_y, on input $\langle d', i, b \rangle$ checks if $i > n$. If so, it rejects. Otherwise, it accepts if and only if $M^A(i) = b$. This machine runs in time $\mathcal{O}(m \log m + \log n)$.

Since f_y is a finite function, it suffices for U to receive the string y as part of the input. Therefore U^A can simulate the computation of M' with oracle A

and f_y by answering the queries to f_y directly from its truth-table. As M' is a two-tape machine it can be simulated by U^A in time $\mathcal{O}(m \log m + \log n)$. Note that $|d'| \leq |d| + \mathcal{O}(\log n)$. Therefore, $\mathrm{KT}^A(x|y) \leq \mathcal{O}(m \log m + \log n)$. □

So far we have considered time-bounded Kolmogorov complexity measures. In Section 5 we will see how these measures give rise to complete problems for time-bounded complexity classes. It is fairly straightforward to define space-bounded notions that will yield complete problems for space-bounded complexity classes.

Definition 3.16 (Space-bounded Kolmogorov complexity). *Let U be a Turing machine and A be an oracle. We define the following three space-bounded complexity measures.*

$$\mathrm{KS}^A_U(x|y) = \min\big\{|d| + s \;\big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } s \text{ space, iff } x_i = b\big\}$$

$$\mathrm{Ks}^A_U(x|y) = \min\big\{|d| + \log s \;\big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } s \text{ space, iff } x_i = b\big\}$$

$$\mathrm{KB}^A_U(x|y) = \min\big\{|d| + 2^s \;\big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } s \text{ space, iff } x_i = b\big\}$$

Again, we denote $x_{|x|+1} = *$. As usual, we omit the superscript A, if $A = \emptyset$ and the string y, if $y = \lambda$.

The notation KB reflects the fact that for a string x, the measure $\mathrm{KB}(x)$ is closely related to the branching program complexity of f_x, the string x viewed as the truth table of a Boolean function. More precisely, if f_x is Boolean function on n inputs and it can be computed by a branching program of size b, then $\mathrm{KB}(x) \leq (b + \log n)^{\mathcal{O}(1)}$. The converse also holds, as every function f_x for which the truth-table has $\mathrm{KB}(x) \leq m$ can be computed by a branching problem of size $(m + \log n)^{\mathcal{O}(1)}$.

Definition 3.17 (Space-optimal machine). *A Turing machine U is*

- KS*-optimal, if for every U', there is a c, s.t.* $\mathrm{KS}^A_U(x|y) \leq c \cdot \mathrm{KS}^A_{U'}(x|y)$ *and if there is a c' such that for every x we have* $\mathrm{KS}_U(x) \leq |x| + c' \log n$,
- Ks*-optimal, if for every U', there is a c, s.t.* $\mathrm{Ks}^A_U(x|y) \leq \mathrm{Ks}^A_{U'}(x|y) + c$ *and if there is a c' such that for every x we have* $\mathrm{Ks}_U(x) \leq |x| + c' \log n$,
- KB*-optimal, if for every U', there is a c, s.t.* $\mathrm{KB}^A_U(x|y) \leq (\mathrm{KB}^A_{U'}(x|y))^c$ *and if there is a c' such that for every x we have* $\mathrm{KB}_U(x) \leq |x| + c' \log n$.

We will fix a universal machine U that is KS-*,* Ks-*, and* KB *optimal and then drop the suffix U when considering the space-bounded Kolmogorov complexity measures.*

We have introduced several K-complexity measures along the same lines as Levin's measure Kt.

Definition 3.18 (Kt**-style measure).** *A Kolmogorov complexity measure* Kμ *is a* Kt-style *measure if the following holds. Given a resource bound r, a universal Turing machine U (possibly nondeterministic or alternating) and a function $f : \mathbb{N} \mapsto \mathbb{N}$,*

$$\mathrm{K}\mu^A_U(x|y) = \min\big\{|d| + f(r) \;\big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts within resource bound } r, \text{ iff } x_i = b\big\} \quad .$$

Again, we denote $x_{|x|+1} = *$.

The Kt-style complexity measures introduced above, just differ in the amount of resources available to the universal machine. It turns out, that this additional power can be "absorbed" into an oracle. For instance, the following fact shows that KT-complexity is essentially KB-complexity relative to an oracle complete for $\mathsf{DTime}[n]$.

Theorem 3.19. *If A is complete for $\mathsf{DTime}[n]$ under $\leq^{\log}_{\mathrm{m}}$ reductions, then there is a constant c, such that* $\mathrm{KT}(x)^{\frac{1}{c}} \leq \mathrm{KB}^A(x) \leq \mathrm{KT}(x)^c$.

Proof. Let A be complete for $\mathsf{DTime}[n]$. Let x be given and let $\mathrm{KT}(x) = m$. Thus, there is a description d_x of length $|d_x| = m$, such that $U(d_x, i) = x_i$ in at most m steps. Consider the set $L_U = \{\langle d_x, i, b\rangle \mid U(d_x, i) = b \text{ in } |d_x| \text{ steps}\}$. Clearly $L_U \in \mathsf{DTime}[n]$, and thus it is many-one reducible to A by reduction f in space $c \log n$. The following algorithm M accepts the input $\langle d_x, i, b\rangle$ if and only if $x_i = b$, and it runs in space $\mathcal{O}(\log |d_x|)$ given oracle A. Given $\langle d_x, i\rangle$, the machine M writes the query $q = f(\langle d_x, i, 1\rangle)$ onto the query tape. If the oracle accepts q then M outputs 1, else, it outputs 0. Clearly, given a correct description d_x, the machine M outputs x_i correctly. The space required for M is $\mathcal{O}(\log|d_x, i, q|) = \mathcal{O}(\log m)$. The universal machine U can simulate M, when given the description $d'_x = \langle M, d_x\rangle$ in space $\mathcal{O}(\log m)$. Further $|d'_x| = m + \mathcal{O}(1)$. Therefore $\mathrm{KB}^A(x) \leq m + 2^{\mathcal{O}(\log m)} = m^{\mathcal{O}(1)}$.

Let $A \in \mathsf{DTime}[n]$ and let x be given, such that $\mathrm{KB}^A(x) = m$. Thus, there is a description d_x of length $|d_x| = m$, such that $U^A(d_x, i) = x_i$ in at most $\log m$ space. Thus $U^A(d_x, i)$ terminates after at most $m^{\mathcal{O}(1)}$ steps. During the computation, U can ask queries of length at most $m^{\mathcal{O}(1)}$, and since $A \in \mathsf{DTime}[n]$, each such query can be answered in time $m^{\mathcal{O}(1)}$. If M denotes the algorithm that simulates the computation of $U^A(d_x, i)$ by directly computing the answers to the oracle queries, then the description $d'_x = \langle M, d_x\rangle$ is sufficient for U to compute $U(d'_x, i) = x_i$ in time $m^{\mathcal{O}(1)}$. As $|d'_x| = m + c$, we can conclude that $\mathrm{KT}(x) \leq m + c + m^{\mathcal{O}(1)} = m^{\mathcal{O}(1)}$. □

We are able to relate the other measures to one another in a similar manner. Most intuitive, perhaps, is the fact that one can interpret the Kt (or the KS) complexity of a string x, as its circuit size relative an oracle complete for E (respectively $\mathsf{DSpace}[n]$). In fact, in this case we get an even tighter relation.

Theorem 3.20. *If A is complete for E under $\leq^{\mathsf{lin}}_{\mathrm{m}}$ reduction, then there is a* KT*-optimal universal Turing machine U and a constant c, such that $\frac{1}{c}\mathrm{Kt}(x) \leq \mathrm{KT}^A_U(x) \leq c \cdot \mathrm{Kt}(x)$.*

If B is complete for $\mathsf{DSpace}[n]$ under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reduction, then there is a KS*-optimal universal Turing machine U and a constant c, such that $\frac{1}{c} \cdot \mathrm{KS}(x) \leq \mathrm{KT}_U^B(x) \leq c \cdot \mathrm{KS}(x)$.*

Proof. The proof is very similar to the proof of Theorem 3.19. Let us prove the relation between Kt and KT^A. The proof for KS is identical.

Let $A \in \mathsf{E}$ and let x be given, such that $\mathrm{KT}^A(x) = m$. Thus, there is a description d_x of length $|d_x| = m$, such that $U^A(d_x, i, b)$ accepts iff $x_i = b$ in at most m time. During the computation, U can ask queries of length at most $\mathcal{O}(m)$, and since $A \in \mathsf{E}$, each such query can be answered in time $2^{\mathcal{O}(m)}$. If M denotes the algorithm that simulates the computation of $U^A(d_x, i, b)$ for every i by directly computing the answers to the oracle queries, then the description $d'_x = \langle M, d_x \rangle$ is sufficient for U to compute $U(d'_x, i, b)$ in time $2^{\mathcal{O}(m)}$. As $|d'_x| = m+c$, we can conclude that $\mathrm{Kt}(x) \leq m+c+\log(2^{\mathcal{O}(m)}) = \mathcal{O}(m)$. Note that this inequality holds regardless of the machine relative to which KT has been defined.

In order to show the other inequality, consider the following. Let A be complete for E. Let x be given and let $\mathrm{Kt}(x) = m$. Let U_t denote the machine relative to which Kt is defined. Consider the set $L_{U_t} = \left\{ \langle d_x, i, b \rangle \;\middle|\; U_t(d_x, i, b) \text{ accepts in } 2^{|d_x|} \text{ steps} \right\}$. Clearly $L_{U_t} \in \mathsf{E}$, and thus it is many-one reducible to A by reduction f in time $c \cdot n$. The following algorithm M accepts the input $\langle d_x, i, b \rangle$ iff $x_i = b$ and runs in time $\mathcal{O}(|d_x|)$ given oracle A. Given $\langle d_x, i, b \rangle$, the machine M writes the query $q = f(\langle d_x, i, b \rangle)$ onto the query tape. If the oracle accepts q then M accepts, else it rejects. Clearly, given a correct description d_x, the machine M accepts $\langle d_x, i, b \rangle$ iff $b = x_i$. The time required for M is $\mathcal{O}(|d_x| + |q|) = \mathcal{O}(m)$. The length of $|d_x|$ is m and thus $\mathrm{KT}_M^A(x) \leq \mathcal{O}(m)$.

Note that the particular choice of M, for instance the number of work tapes of M, depends on A. Let U_T be the machine relative to which KT is defined. Since U_T is a KT optimal universal machine we can define the machine U that on input $\langle 1, d \rangle$ runs $U_T(d)$ and on input $\langle 0, d \rangle$ runs $M(d)$. This machine is trivially a KT-optimal

universal machine and yet it maintains a linear relation between $\mathrm{KT}^A_U(x)$ and $\mathrm{Kt}(x)$. Furthermore, in the first part we argued that $\mathrm{Kt}(x) = \mathcal{O}(\mathrm{KT}_{M'}(x))$, relative to any machine M'. Therefore the machine U satisfies Remark 3.14 as $\mathrm{KT}^A_U(x) \leq c'\mathrm{KT}^A_{M'}(x)$. □

As the different complexity measures considered can be captured by KB relative to different oracles, it seems natural that the question whether two such measures differ substantially is directly related to the question if certain complexity classes coincide. For instance, KT and Kt are polynomially related (i.e. for all x, $\mathrm{KT}(x) = \mathrm{Kt}(x)^{\mathcal{O}(1)}$), if and only if $\mathsf{EXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$. This type of question is being investigated in more depth in Section 4 (→Theorem 4.6).

All the Kt-style complexity measures were defined in terms of a deterministic universal Turing machine. It is possible to define similar measures using nondeterministic or alternating machines with the various resource bounds. We will focus on two measures for which we are able to show interesting results.

The measure KNt is a nondeterministic version of Kt. We will later show this measure to be complete for $\mathsf{NEXP}/_{\mathsf{poly}}$. It is the most restrictive Kt-style measure for which we can show unconditionally that we cannot recognize strings with high complexity in polynomial time.

Definition 3.21. *Let U be a nondeterministic Turing machine and A be an oracle.*

$$\mathrm{KNt}^A_U(x|y) = \min\big\{|d| + \log t \;\big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } t \text{ steps, iff } x_i = b\big\}$$

We denote $x_{|x|+1} = *$. As usual, we omit the superscript A, if $A = \emptyset$ and the string y, if $y = \lambda$.

We will consider a nondeterministic universal machine U to be KNt-optimal, if for every machine U' there is a constant c, such that $\mathrm{KNt}^A_U(x|y) \leq \mathrm{KNt}^A_{U'}(x|y) + c$. Since

there is a universal machine that can simulate any other nondeterministic machine with linear overhead ([BG70]), we know that there is a KNt-optimal universal machine.

In general, it seems that a nondeterministic machine is less powerful than a deterministic machine with a nondeterministic oracle. Thus it might seem that if A is an oracle complete for NE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$, then for some strings x, $\mathrm{KT}^A(x)$ could be significantly less than $\mathrm{KNt}(x)$. However, this intuition turns out to be false. There is a linear relation between those two measures. This is due to the fact that $\mathsf{NE}/_{\mathsf{lin}}$ is closed under complement (→Lemma 5.34).

Theorem 3.22. *If A is complete for NE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reduction, then there is a* KNt*-optimal machine U and a constant c, such that $\frac{1}{c} \cdot \mathrm{KNt}(x) \leq \mathrm{KT}^A(x) \leq c \cdot \mathrm{KNt}(x)$.*

Proof. Consider the first inequality. Let $A \in \mathsf{NE}$. and let x be given, such that $\mathrm{KT}^A(x) = m$. Thus, there is a description d_x of length $|d_x| \leq m$, such that $U^A(d_x, i) = x_i$ in at most m time. During the computation, U can ask queries of length at most $\mathcal{O}(m)$. Consider the set $A' = \{u1a \mid a \in A \text{ and } u \in 0^*\}$. Clearly, the computation $U^A(d_x, i)$ can be performed by an algorithm U' relative to A', such that all queries for all $1 \leq i \leq |x|$ are of length $c \cdot m$ for some constant c.

Note that $A' \in \mathsf{NE}$. Thus there is a deterministic machine M_1, such that if $q \in A'$, then there is a $w \in \{0,1\}^{2^n}$ for which $M_1(q, w)$ accepts in time $\mathcal{O}(2^{|q|})$. Lemma 5.34 yields $A' \in \mathsf{co\text{-}NE}/_{\mathsf{lin}}$. Thus there is an advice function $a : \mathbb{N} \mapsto \{0,1\}^*$ with $|a(n)| = \mathcal{O}(n)$ and a machine M_2, such that if $q \notin A'$, if and only if there is a $w \in \{0,1\}^{2^n}$ for which $M_2(a(|q|), q, w)$ accepts.

Let M denote the following nondeterministic algorithm. Let a be the advice function discussed above. Recall that all queries are of length $c \cdot m$. On input $d' = \langle d_x, i, b, a(c \cdot m)\rangle$, M simulates the computation of $U'^{A'}(d_x, i, b)$. It directly computes the answer to each oracle query q, by guessing a positive witness w_1 and a

negative witness w_2. The machine M then computes $M_1(q, w_1)$ and $M_2(a(|q|), q, w_2)$. If M_1 accepts then $q \in A'$. If M_2 accepts, then $q \notin A'$. If neither machine accepts, then the guess of the witnesses was wrong and the particular computation path of the nondeterministic machine aborts. All paths that do not abort the computation accept, if and only if the simulated computation $U'^{A'}(d_x, i, b)$ accepts. It is straightforward to verify that there is always a computation path that correctly simulates the computation of $U'^{A'}$, and that all paths that do not correctly simulate A' recognize this and abort the computation. The run-time required to compute the answer to an oracle query requires at most $2^{\mathcal{O}(m)}$ steps. Thus the run-time of the overall computation can be bounded by $2^{\mathcal{O}(m)} \cdot 2^{\mathcal{O}(m)} = 2^{\mathcal{O}(m)}$. Therefore $\mathrm{KNt}_M(x) \leq |d'| + \log 2^{\mathcal{O}(m)} = \mathcal{O}(m)$. As KNt is defined relative to a KNt-optimal machine we have $\mathrm{KNt}(x) \leq c' \cdot \mathrm{KNt}_M(x) = \mathcal{O}(m)$.

Now consider the second inequality. Let A be complete for NE. Let x be given and let $\mathrm{KNt}(x) = m$. Let U_N be the KNt-optimal machine relative to which KNt is defined. Thus, there is a description d_x of length $|d_x| = m$, such that the nondeterministic universal machine U_N accepts the input $\langle d_x, i, b\rangle$ in at most 2^m steps, if and only if $x_i = b$. Consider the set $L_{U_N} = \left\{ \langle d_x, i, b\rangle \;\middle|\; U_N(d_x, i, b) \text{ accepts in } 2^{|d_x|} \text{ steps} \right\}$. Clearly $L_{U_N} \in \mathsf{NE}$, and thus it is many-one reducible to A by reduction f in time $c \cdot n$. The following algorithm M accepts the input $\langle d_x, i, b\rangle$ if and only if $b = x_i$, and it runs in time $\mathcal{O}(|d_x|)$ given oracle A. Given $\langle d_x, i, b\rangle$, the machine M writes the query $q = f(\langle d_x, i, b\rangle)$ onto the query tape. If the oracle accepts q then M accepts, otherwise it rejects. Clearly, given a correct description d_x, the machine M accepts $\langle d_x, i, b\rangle$ if and only if $b = x_i$. The time required for M is $\mathcal{O}(|d_x| + |q|) = \mathcal{O}(m)$. Further $|d'_x| = m + \mathcal{O}(1)$ and thus $\mathrm{KT}^A_M(x) \leq \mathcal{O}(m)$.

Note that the particular choice of M, for instance the number of work tapes of M, depends on A. Let U_T be the machine relative to which KT is defined. Since U_T is a KT optimal universal machine we can define the machine U that on input $\langle 1, d\rangle$

runs $U_T(d)$ and on input $\langle 0, d\rangle$ runs $M(d)$. This machine is trivially a KT-optimal universal machine and yet it maintains a linear relation between $\mathrm{KT}^A_U(x)$ and $\mathrm{KNt}(x)$. Furthermore, in the first part we argued that $\mathrm{Kt}(x) = \mathcal{O}(\mathrm{KT}_{M'}(x))$, relative to any machine M'. Therefore the machine U satisfies Remark 3.14 as $\mathrm{KT}^A_U(x) \leq c'\mathrm{KT}^A_{M'}(x)$.

□

We introduce a Kt-style complexity measure that is based on an alternating universal machine. While we observed a close connection between circuit size of a function and the KT complexity of its characteristic string, as well as a connection between branching program size and KB complexity, this new measure KF yields a connection to the formula size of a function. It possible to view a Boolean formula as a special case of a Boolean circuit. A formula is a circuit in which every gate, except the input gates, has fan-out 1. The size of a Boolean formula is thus the size of of the fan-out 1 circuit representing the formula.

Definition 3.23. *Let U be an alternating Turing machine and A be an oracle.*

$$\mathrm{KF}^A_U(x|y) = \min\Big\{|d| + 2^t \;\Big|\; \forall b \in \{0,1,*\}\ \forall i \leq |x|+1 :\ U^A(d,i,y,b) \text{ accepts in } t \text{ steps, iff } x_i = b\Big\}$$

We denote $x_{|x|+1} = *$. As usual, we omit the superscript A, if $A = \emptyset$ and the string y, if $y = \lambda$.

We will consider an alternating universal machine U to be KF-optimal, if for every machine U' there is a constant c, such that $\mathrm{KF}^A_U(x|y) \leq (\mathrm{KF}^A_{U'}(x|y))^c$. Again, the existence of such an optimal machine, follows from standard simulation constructions. The definitions of the complexity measures introduced directly implies an ordering on the measures.

Definition 3.24 (Implicit order on K-complexity measures). *We consider the following order on the* Kt*-style measures considered.*

$$\mathrm{K} \leq \mathrm{Ks} \leq \mathrm{KNt} \leq \mathrm{Kt} \leq \mathrm{KS} \leq \mathrm{KT} \leq \mathrm{KB} \leq \mathrm{KF} \quad .$$

This order is motivated by the following fact.

Fact 3.25. *The following relations hold for every x and a fixed universal constant c.*

(i) $\mathrm{K}(x) \leq \mathrm{Kt}(x) \leq \mathrm{KT}(x)$,

(ii) $\mathrm{K}(x) \leq \mathrm{Ks}(x) \leq \mathrm{KS}(x) \leq \mathrm{KB}(x)$,

(iii) $\mathrm{KB}(x) \leq (\mathrm{KF}(x))^c$,

(iv) $\mathrm{KT}(x) \leq (\mathrm{KB}(x))^c$,

(v) $\mathrm{Kt}(x) \leq c \cdot \mathrm{KS}(x)$.

(vi) $\mathrm{Ks}(x) \leq c \cdot \mathrm{KNt}(x)$.

Proof. Items (i) and (ii) follow immediately form the definitions. Items (iii), (iv), (v), and (vi) can be proven directly by using the standard simulation of an $s(n)$ space-bounded machine with a $2^{\mathcal{O}(s(n))}$ time-bounded machine. □

All the K-complexity measures considered so far measure the length of descriptions that are sufficient to generate the string of interest. It is possible to modify this point of view and consider descriptions that are sufficient to recognize the string of interest when it provided as input. In this case, we will say that the description d_x is sufficient to *distinguish* the string x.

For the resource-unbounded case (or for sufficiently generous, fixed resource bounds), these two points of view yield essentially equivalent measures. Certainly, if a string x

can be generated from a description d_x, that same description is sufficient to distinguish x. (Simply generate x from d_x and then only accept an input z if $z = x$.) On the other hand if it is possible to distinguish x given a description d_x, it is possible to generate x by enumerating all string of length $|x|$ and determine which one is being distinguished by description d_x.

This notion of distinguishing complexity was first introduced by Sipser in [Sip83], where he considered the measure in a fixed polynomial time-bounded setting. Distinguishing complexity has since been subject to further study, e.g. in [BFL02]. We transfer this notion of complexity to the context of Kt-style measures and obtain the measure KDt.

Definition 3.26 (KDt)**.** *Let U be a Turing machine and let A be an oracle. Define the measure* $\mathrm{KDt}^A_U(x|y)$ *of a string x as*

$$\mathrm{KDt}^A_U(x|y) = \min\big\{|d| + \log t \;\big|\; U^A(d, z, y) \text{ runs in time } 2^t \text{ for all } z \in \{0,1\}^{|x|} \text{ and } U^A(d, z, y) \text{ accepts, if and only if } z = x\big\}$$

We use $\mathrm{KDt}_U(x|y)$ *to denote* $\mathrm{KDt}^{\emptyset}_U(x|y)$*, and we use* $\mathrm{KDt}_U(x)$ *to denote* $\mathrm{KDt}_U(x|\lambda)$*.*

As for the other complexity measures, we will mostly consider the KDt measure relative to an optimal machine.

Definition 3.27 (KDt**-optimal).** *A universal machine U is* KDt-optimal*, if for every Turing machine U' there is a constant c for which the following holds.*

$$\mathrm{KDt}_U(x) \leq \mathrm{KDt}_{U'}(x) + c\log|x| \quad .$$

We will fix a KDt*-optimal Turing machine U and use* $\mathrm{KDt}(x)$ *to denote* $\mathrm{KDt}_U(x)$*.*

Note that the definition yields that there is constant c such that for all x the measure $\mathrm{KDt}(x) \leq |x| + c\log|x|$.

3.5 Symmetry of Information

Given two strings x and y, one can consider how much information y has about x. Formally, this can be captured with the measure $\mathrm{I}(x : y) = \mathrm{K}(x) - \mathrm{K}(x|y)$, the *information of x about y*. In [ZL70] it was shown that this measure is symmetric: $\mathrm{I}(x : y) \approx \mathrm{I}(y : x)$. That is, for any two strings, the first string carries roughly the same amount of information about the second string, as the second about the first. The proof of [ZL70] actually shows that a stronger statement holds:

Theorem 3.28. *Given two strings x and y, there is a constant c, such that*

$$\mathrm{K}(x) + \mathrm{K}(y|x) - c\log|xy| \;\leq\; \mathrm{K}(xy) \;\leq\; \mathrm{K}(x) + \mathrm{K}(y|x) + c\log|xy| \quad .$$

This statement is usually referred to in the literature as *symmetry of information* [LV97, For01]. In the proof of Theorem 3.31 we reiterate the argument that symmetry of information implies $\mathrm{K}(x)+\mathrm{K}(y|x) \approx \mathrm{K}(y)+\mathrm{K}(x|y)$ and thus x and y carry roughly the same information content. In order to distinguish the latter statement from the former, Lee and Romashchenko [LR04] denote this statement as *symmetry of mutual information*.

The question of whether or not symmetry of information holds in different settings, has repeatedly been studied in the literature. Longpré showed in his thesis [Lon86] (also with Mocas in [LM93]), that symmetry of information holds in a sense for resource-bounded complexity measures with at least exponential time bounds or polynomial space bounds. In [LW95], Longpré and Watanabe gave evidence that the question of symmetry of information for polynomial time-bounded K-complexity is unlikely to be easily resolved. If it holds, then one-way functions cannot exist. If it does not hold, then $\mathsf{P} \neq \mathsf{NP}$.

In all cases in the literature that establish symmetry of information for a resource-bounded complexity measure, the statement is not as intuitive to interpret as in the

resource-unbounded case. All these cases mentioned above consider symmetry of information for K-complexity measures with fixed resource bounds – as opposed to Kt-style measures that consider the resource bound as part of the measure. However, these theorems establishing symmetry of information are all of the form "For every (say, exponential) time bound t, there is an (exponential) time bound t', such that $\mathrm{K}^{t'}_{\mathrm{t}}(x) + \mathrm{K}^{t'}_{\mathrm{t}}(y|x) \leq \mathrm{K}^{t}_{\mathrm{t}}(xy)$." Thus, when only considering measures with *identical* time-bounds, symmetry of information is not established by these results.

So far, there are not any unconditional results establishing that symmetry of information does not hold for any resource-bounded K-complexity measure. There are oracle constructions, relative to which symmetry of information does not hold ([BF95, LR04]). However, when we consider sufficiently powerful Kt-style complexity measures, we show in this section that symmetry of information does not hold for these.

3.5.1 Definition of Symmetry of Information

As stated before, there is some ambiguity in the literature with regard to the term *symmetry of information*. Its origin stems from the relation $\mathrm{I}(x : y) = \mathrm{I}(y : x) \pm \mathcal{O}(\log|xy|)$, which can be directly expressed in terms of Kolmogorov complexity as $\mathrm{K}(x) + \mathrm{K}(y|x) = \mathrm{K}(y) + \mathrm{K}(x|y) \pm \mathcal{O}(\log|xy|)$. However, all proofs in the literature establishing this prove the following stronger relation, which we will use to define symmetry of information.

Definition 3.29 (Symmetry of information). *Let* $\mathrm{K}\mu$ *be a* K*-complexity measure. We say* symmetry of information *holds for* $\mathrm{K}\mu$*, if there is a constant* $c > 0$ *such that*

$$\mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) - c\log(|xy|) \leq \mathrm{K}\mu(xy) \leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + c\log(|xy|)$$

for every $x, y \in \{0,1\}^n$ *and all large* n*.*

Following the terminology used in [LR04], we define the slightly weaker condition of symmetry of mutual information.

Definition 3.30 (Symmetry of mutual information). *Let* $\mathrm{K}\mu$ *be a* K*-complexity measure. We say* symmetry of mutual information *holds for* $\mathrm{K}\mu$*, if there is a constant* $c > 0$ *such that*

$$\mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) \leq \mathrm{K}\mu(y) + \mathrm{K}\mu(x|y) + c\log(|xy|)$$

for all $x, y \in \{0,1\}^n$ *all large* n*.*

As mentioned above, for sufficiently powerful K-complexity measures, symmetry of information implies symmetry of mutual information.

Fact 3.31 (Symmetry of mutual information). *Let* $\mathrm{K}\mu$ *be a* Kt*-style complexity measure for which* $\mathrm{K}\mu \geq \mathrm{KS}$*. If symmetry of information holds for* $\mathrm{K}\mu$*, then symmetry of mutual information holds for* $\mathrm{K}\mu$*.*

Proof. First note that we can generate xy, by first generating x and then y given x. Thus $\mathrm{K}\mu(xy) \leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + c_1\log(|xy|)$. Furthermore, the concatenation yx can be obtained from xy, given the length of x. Therefore $\mathrm{K}\mu(yx) \leq \mathrm{K}\mu(xy) + c_2\log|xy|$. Let us assume that symmetry of information holds for y and x, i.e. there is a constant $c_3 > 0$ such that

$$\mathrm{K}\mu(y) + \mathrm{K}\mu(x|y) - c_3\log(|yx|) \leq \mathrm{K}\mu(yx) \quad .$$

With the two observations above we can conclude

$$\begin{aligned} \mathrm{K}\mu(y) + \mathrm{K}\mu(x|y) - c_3 \log|yx| &\leq \mathrm{K}\mu(yx) \\ &\leq \mathrm{K}\mu(xy) + c_2 \log|xy| \\ &\leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + (c_1 + c_2) \log|xy| \quad . \end{aligned}$$

Since $|xy| = |yx|$ we can pick $c = c_1 + c_2 + c_3$ and obtain

$$\mathrm{K}\mu(y) + \mathrm{K}\mu(x|y) \leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + c \log|xy| \quad .$$

□

The restriction to strings of length n is just for matters of convenience. We can show the following more general fact.

Fact 3.32. *Symmetry of information holds for a* K*-complexity measure* $\mathrm{K}\mu \geq \mathrm{KB}$*, if there is a constant* c*, such that*

$$\mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) - c \log(|xy|) \leq \mathrm{K}\mu(xy) \leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + c \log(|xy|)$$

for all $x, y \in \{0,1\}^*$.

Proof. The proof is a straightforward padding argument. We have to show that symmetry of information for strings of equal length, implies symmetry of information of strings of arbitrary length.

Let x and y be given such that $|x| = n$ and $|y| = m$. Without loss of generality, let $n > m$. Consider $y' = 0^{n-m-1}1y$. Clearly $|y'| = n$. Using symmetry of information for strings of equal length, we can conclude that there is a constant c_1, such that

$$\mathrm{K}\mu(xy') \leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y'|x) + c_1 \log(|xy'|) \quad .$$

As y' is easily derived from y, there is a constant c_2 such that

$$\mathrm{K}\mu(xy) \leq \mathrm{K}\mu(xy') + c_2 \log|xy'|$$

and a constant c_3 such that

$$\mathrm{K}\mu(y'|x) \leq \mathrm{K}\mu(y|x) + c_3 \log|xy'| \quad .$$

Therefore

$$\begin{aligned}
\mathrm{K}\mu(xy) &\leq (\mathrm{K}\mu(x) + \mathrm{K}\mu(y'|x) + c_1 \log(|xy'|)) + c_2 \log|xy'| \\
&\leq \mathrm{K}\mu(x) + (\mathrm{K}\mu(y|x) + c_3 \log(|xy'|)) + (c_1 + c_2) \log|xy'| \\
&= \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + (c_1 + c_2 + c_3) \log|xy'| \quad ,
\end{aligned}$$

As $\log|xy'| = \mathcal{O}(\log|xy|)$ we obtain the desired

$$\mathrm{K}\mu(xy) \leq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) + c_4 \log|xy|$$

for some constant c_4. Similarly we can argue that $\mathrm{K}\mu(xy) \geq \mathrm{K}\mu(x) + \mathrm{K}\mu(y|x) - c_4 \log|xy'|$. Thus $\mathrm{K}\mu(xy)$ is within additive logarithmic terms of $\mathrm{K}\mu(x) + \mathrm{K}\mu(y|x)$. □

3.5.2 Symmetry of Information for Kt

In this subsection we will prove that symmetry of information does not hold for any sufficiently powerful Kt-style complexity measure. In particular, it does not hold for Kt itself.

Theorem 3.33. *Symmetry of information does not hold for* Kt*.*

Proof. We will show that for every c and every sufficiently large n, there are two

strings $x, y \in \{0,1\}^n$, such that

$$\mathrm{Kt}(x) > \tfrac{1}{2}n \qquad \text{and} \qquad \mathrm{Kt}(y|x) > \tfrac{1}{2}n$$

but

$$\mathrm{Kt}(xy) \leq \tfrac{1}{2}n + \mathcal{O}(\log n)\ .$$

We obtain x and y by a simple diagonalization. We will define x as the output of the following machine M.

Algorithm 3.34 (M).

```
(1)  input:  1^n;
(2)      W := ∅;
(3)      forall i ≤ n/2
(4)          forall d ∈ {0,1}^i
(5)              w := U(d) after 2^(n/2 − i) steps;
(6)              W := W ∪ {w};
(7)          next d;
(8)      next i;
(9)      sort W
(10)     x := 0^n;
(11)     while x ∈ W do
(12)         x := lexicographic successor of x;
(13)     end-while;
(14) output x;
```

Note that M generates a list W of *all* strings w of length n with $\mathrm{Kt}(w) \leq \frac{1}{2}n$. Since x is picked so it does not belong to W, we have $\mathrm{Kt}(x) > \frac{1}{2}n$.

The simulation of U in Line `(5)` requires $2^{\frac{n}{2}-i}$ steps. Adding w to W requires $\mathcal{O}(n)$ steps. Repeating this for each description of length i results in a run-time of $2^i \cdot (2^{\frac{n}{2}-i} + \mathcal{O}(n)) \leq 2^{\frac{n}{2}+\mathcal{O}(\log n)}$. This segment is repeated $\frac{n}{2}$ times and thus the run time for it can still be bounded from above by $2^{\frac{n}{2}+\mathcal{O}(\log n)}$. Sorting W in Line `(9)` requires $\mathcal{O}(|W| \log |W|) = \mathcal{O}(2^{\frac{n}{2}} \log 2^{\frac{n}{2}}) \leq 2^{\frac{n}{2}+\mathcal{O}(\log n)}$. The `while`-loop from Line `(11)` to Line `(13)` requires at most $2^{\frac{n}{2}}$ iterations. In each iteration the test $w \in W$ requires

at most $\mathcal{O}(n)\cdot\log 2^{\frac{n}{2}} = \mathcal{O}(n)$ steps. This results in a runtime of $2^{\frac{n}{2}}+\mathcal{O}(n) \leq 2^{\frac{n}{2}+\mathcal{O}(\log n)}$ steps for the loop. Thus the overall number of steps the machine M runs on input 1^n is bounded by $2^{\frac{n}{2}+\mathcal{O}(\log n)}$. Given a description for M and n, the universal machine can thus generate x in $2^{\frac{n}{2}+\mathcal{O}(\log n)}\cdot\mathcal{O}(\log(2^{\frac{n}{2}+\mathcal{O}(\log n)})) \leq 2^{\frac{n}{2}+\mathcal{O}(\log n)}$ steps from a description d_x of size $c + \log n$.

With a similar argument we can construct a string y, such that $\mathrm{Kt}(y|x) > \frac{1}{2}n$. But there is a description d_y of size $\mathcal{O}(1)$, such that $U(d_y, x) = y$ in $2^{\frac{n}{2}+\mathcal{O}(\log n)}$ steps.

In order to generate the string xy, the universal machine uses the description d_x to output x in $2^{\frac{n}{2}+\mathcal{O}(\log n)}$ steps. Then it uses description d_y and x to generate y in $2^{\frac{n}{2}+\mathcal{O}(\log n)}$ steps. Hence

$$\begin{aligned}\mathrm{Kt}(xy) &\leq |\langle d_x, d_y\rangle| + \log(2^{\frac{n}{2}+\mathcal{O}(\log n)} + 2^{\frac{n}{2}+\mathcal{O}(\log n)}) \\ &\leq \log(|d_x|) + |d_x| + |d_y| + \tfrac{1}{2}n + \mathcal{O}(\log n) + 1 \\ &\leq \tfrac{1}{2}n + \mathcal{O}(\log n) \quad .\end{aligned}$$

On the other hand

$$\mathrm{Kt}(x) + \mathrm{Kt}(x|y) > \tfrac{1}{2}n + \tfrac{1}{2}n = n \quad .$$

Thus symmetry of information does not hold for Kt. □

Note that in the above proof the crucial resource requirement is the ability to enumerate all possible short descriptions. For each Kt-style measure $\mathrm{K}\mu \geq \mathrm{Kt}$ the universal machine has sufficient resources to run the above algorithm. Therefore we obtain the following corollary.

Corollary 3.35. *Let* $\mathrm{K}\mu$ *be any resource-bounded* Kt*-style complexity measure with* $\mathrm{K}\mu \geq \mathrm{Kt}$*. Symmetry of information does not hold for* $\mathrm{K}\mu$*.*

4 Simple Strings in a Set

In Section 3 several different Kt-style measures were considered. The measures were all defined in a similar manner and essentially only vary in the computational resources available to the machine decoding the descriptions. It is not surprising that the question of whether or not any of these measures are closely (say polynomially) related is closely linked to the question of whether or not additional resources necessarily make computations more powerful.

For instance, one could ask the question: what would be implied, if Kt and KT are polynomially related? Recall that we can interpret $\mathrm{KT}(x)$ as an approximation of the circuit size required to compute the function f_x having truth-table x. Likewise, $\mathrm{Kt}(x)$ approximates the circuit size of f_x on circuits that have access to an oracle complete for E. Then the observation that $\mathrm{Kt}(x)$ and $\mathrm{KT}(x)$ are polynomially related would imply that polynomial size circuits do not gain significant power from an oracle from E. Namely, it would imply $\mathsf{P}/_{\mathsf{poly}} = \mathsf{P}^{\mathsf{E}}/_{\mathsf{poly}}$.

4.1 Preliminaries

As stated before, a polynomial relation between different Kt-style complexity measures implies several different statements in complexity. In this subsection we will provide the terminology to present these observations.

One of the main observations is, that a polynomial relation between different Kt-style complexity measures are directly related to the question of how hard it is to describe the most simple strings in a set. In order to measure the complexity of the simple strings, we will extend the Kt-style complexity measure to sets as follows.

Definition 4.1 ($\mathrm{K}\mu_L$)**.** *Let* $\mathrm{K}\mu$ *be a* Kt*-style measure and let* L *be a set. Define*

$$\mathrm{K}\mu_L(n) = \min\{\mathrm{K}\mu(x) \mid |x| = n \text{ and } x \in L\} \quad .$$

If $L^{=n} = \emptyset$*, then* $\mathrm{K}\mu_L(n)$ *is undefined.*

Some of the following theorems will talk about certain types of sets. We will use the following (standard) terminology.

Definition 4.2.

(i) A set L *is* dense*, if* $\mathrm{cens}_L(n) \geq \frac{2^n}{n^k}$ *for some* $k \geq 0$*.*

(ii) A set L *is* P-printable*, if there a machine* M *running in polynomial time, that on input* 1^n *outputs all strings of length* n *in* L*.*

In order to state the following theorems as strong as possible we will use the following definition.

Definition 4.3 (Witness). *Let* L *be a set decidable by a nondeterministic machine* N *within time-bound* t*. A string* w *is a witness for* x *with respect to* N*, if* $|w| = t(|x|)$*,* $x \in L$ *and* $N(x)$ *has an accepting computation path, where the* i*th nondeterministic choice corresponds to the* i*th bit of* w*.*

The complexity of searching for witnesses is related to complexity of the simplest string for sets in P. In order to formalize that more precisely we will use the following terminology.

Definition 4.4 (NEXP-search problem). *Given a nondeterministic machine* N *running in time* 2^{n^c}*, we define a* NEXP*-search problem as the problem: given* x*, if* $x \in L(N)$*, find a witness* w *for* x*.*

A NEXP*-search problem is solvable in exponential time, if there is a machine* M *that on input* x *outputs a witness* w*, if* $x \in L$*.*

A NEXP*-search problem is solvable by polynomial size circuits, if there is a family of circuits* $(C_i)_{i \in \mathbb{N}}$ *with the following property: for every string* x *of length* n *if* $x \in L(N)$*, then the string* $w = C_n(x,1)C_n(x,2)\ldots C_n(x,2^{|x|^c})$ *is a witness for* x*.*

Along similar lines we will define the notion of FewEXP-search problems.

Definition 4.5 (FewEXP-search problem). *Given a nondeterministic machine N running in time 2^{n^c}, we define a* FewEXP*-search problem as the problem: given x, if there are at most $2^{|x|^c}$ witnesses for x with respect to N, find a witness w if $x \in L(N)$. We define a* FewEXP *decision problem as the problem: given x, if there are at most $2^{|x|^c}$ witnesses for x with respect to N, output 1 if $x \in L(N)$.*

A FewEXP*-search problem is solvable in exponential time, if there is a machine M that runs in exponential time and on input x outputs a witness w for x with respect to N, if $x \in L(N)$ and x has at most $2^{|x|^c}$ such witnesses.*

A FewEXP*-search problem is solvable by polynomial size circuits, if for every nondeterministic machine running in time $2^{|x|^c}$ there is a family of circuits $(C_i)_{i\in\mathbb{N}}$ with the following property: for every string x of length n, if there are at most $2^{|x|^c}$ witnesses for x and $x \in L(N)$, then the string $w = C_n(x,1)C_n(x,2)\dots C_n(x,2^{|x|^c})$ is a witness for w.*

A FewEXP*-decision problem is solvable in exponential time (by polynomial size circuits), if there is a deterministic machine M running in exponential time (a family of polynomial size circuits $(C_i)_{i\in\mathbb{N}}$), such that $M(x)$ (reps. $C(x)$) rejects if x does not have any witnesses with respect to N, and $M(x)$ (respectively $C_{|x|}(x)$) accepts if $N(x)$ accepts, but x has at most $2^{|x|^c}$ witnesses with respect to N. If x has more than $2^{|x|^c}$ the behavior of M (of C) can be arbitrary.*

4.2 Simple strings for sets in P

In this subsection we will argue, that the complexity of the most simple strings for certain sets in P has a direct impact on the hardness of certain complexity classes.

Theorem 4.6 ([All01]). *The following statements are equivalent.*

(i) *For every string* x, $\mathrm{KT}(x) = (\mathrm{Kt}(x) + \log|x|)^{\mathcal{O}(1)}$.

(ii) $\mathsf{EXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$.

(iii) *For every* P*-printable set* $L \in \mathsf{P}$, $\mathrm{KT}_L(n) = (\log n)^{\mathcal{O}(1)}$.

(iv) *There is a dense set* $L \in \mathsf{P}/_{\mathsf{poly}}$, *such that* $\mathrm{Kt}_L(n) > n^{\varepsilon}$ *for some* ε.

Since we will present theorems similar to Theorem 4.6, we will present the proof sketched in [All01].

Proof of Theorem 4.6. We will prove the implications (ii) $\implies$ (i) $\implies$ (iii) $\implies$ (ii), as well as the equivalence (iv) $\iff$ (ii)

(ii) $\implies$ (i)

Let x be a string with $\mathrm{Kt}(x) = m$ for some m. Thus there is a machine M and a description d of length m, such that $M(d) = x$ in time $2^{|d|}$. Consider the set $L_M = \big\{\langle d, i\rangle \;\big|\; M(d) \text{ runs in time } 2^{|d|} \text{ and the } i\text{th bit of } M(d) \text{ is } 1\big\}$. Clearly $L_M \in \mathsf{EXP}$ and by the assumption that $\mathsf{EXP} \in \mathsf{P}/_{\mathsf{poly}}$, we can conclude that there is a circuit C of size m^c for some c, such that $C(d, i) = x_i$. If we hard-code d and $|x|$ into the circuit, we obtain a circuit C' of size $\mathcal{O}(m^c)+\log|x|$ computing f_x - the function with x as its truth-table. Now we can apply Theorem 3.15 and conclude that $\mathrm{KT}(x) \leq m^c + \log|x|^{\mathcal{O}(1)} = (\mathrm{Kt}(x) + \log|x|)^{\mathcal{O}(1)}$.

(i) $\implies$ (iii)

Let L be a P-printable set. Thus, there is a constant c and a machine M_L, such that in n^c steps $M_L(1^n) = \langle y_1, \ldots, y_k\rangle$ where $k \leq n^c$ and $L^{=n} = \{y_1, \ldots, y_k\}$. So given description $d = \langle M_L, i\rangle$ there is an algorithm that outputs y_i in $\mathcal{O}(n^c)$

steps. Therefore, each y_i has low Kt-complexity:

$$\begin{aligned}\mathrm{Kt}(y_i) &\leq |\langle M_L, n, i\rangle| + \log(n^c) \\ &\leq (\log n)^{c'} \quad ,\end{aligned}$$

for some constant c'. By assumption KT and Kt are polynomially related. Therefore, $\mathrm{KT}(y_i) \leq (\mathrm{Kt}(y_i))^c \leq (\log n)^{\mathcal{O}(1)}$.

(iii) $\Longrightarrow$ (ii)
Let $f \in \mathsf{E}$. Consider the set

$$\begin{aligned}L_f = \{x \;\big|\; |x| = 2^k, \text{ for some } k \text{ and } x \text{ is the truth-table} \\ \text{of } f \text{ for inputs of size } k\} \quad .\end{aligned}$$

Note that L_f is P-printable. On input $n = 2^k$ the machine printing L_f computes $f(z)$ for all inputs z of length k. This can be done in time $2^k \cdot 2^{\mathcal{O}(k)} = n^{\mathcal{O}(1)}$.

By assumption $x = \chi_f \in L_f$ has low KT complexity. Namely, $\mathrm{KT}(x) \leq \log |x|^{\mathcal{O}(1)}$. As the KT-complexity of the truth-table χ_f of a function f is polynomially related to the circuit size of f ($\rightarrow$Theorem 3.15), we can conclude that $\mathrm{SIZE}(f) \leq n^{\mathcal{O}(1)}$. This shows that $\mathsf{E} \subseteq \mathsf{P}/_{\mathsf{poly}}$. With a standard padding argument one can see that this also implies $\mathsf{EXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$.

(iv) $\Longrightarrow$ (ii)
The authors of [BFNW93] essentially prove the contrapositive of this statement. They argue, if $\mathsf{EXP} \not\subseteq \mathsf{P}/_{\mathsf{poly}}$ then the output of their pseudo-random generator G_f^{BFNW} cannot be distinguished from truly random strings by any set in $\mathsf{P}/_{\mathsf{poly}}$. This implies that every dense set in $\mathsf{P}/_{\mathsf{poly}}$ must contain a certain fraction of simple strings.

This statement also follows immediately from the proof of Theorem 5.16. If any dense set $L \in \mathsf{P}/_{\mathsf{poly}}$ contains only strings x with $\mathrm{Kt}(x) \geq |x|^{\varepsilon}$, it follows that $\mathsf{EXP} \subseteq \mathsf{P}^{L}/_{\mathsf{poly}} = \mathsf{P}/_{\mathsf{poly}}$.

(ii) $\Longrightarrow$ (iv)
The argument used to prove this implication is due to [ISW99]. Assume that every dense set in $\mathsf{P}/_{\mathsf{poly}}$ contains a string x with $\mathrm{Kt}(x) \leq |x|^{\varepsilon}$. Thus the set $R'_{\mathrm{Kt}} = \left\{x \;\middle|\; \mathrm{Kt}(x) > |x|^{\varepsilon}\right\}$ does not belong to $\mathsf{P}/_{\mathsf{poly}}$. However, it is easy to see that R'_{Kt} belongs to EXP (just enumerate all descriptions of length $|x|^{\varepsilon}$ and determine if one them yields x in $2^{|x|^{\varepsilon}}$ steps). Thus $\mathsf{EXP} \not\subseteq \mathsf{P}/_{\mathsf{poly}}$. □

Theorem 4.7. *The following statements are equivalent.*

(i) *For every string x,* $\mathrm{KT}(x) = (\mathrm{KDt}(x) + \log|x|)^{\mathcal{O}(1)}$.

(ii) FewEXP*-decision problems can be solved with polynomial circuits.*

(iii) FewEXP*-search problems can be solved with polynomial-size circuits.*

(iv) *For every $L \in \mathsf{P}$,* $\mathrm{KT}_L(n) = (\log|L^{=n}| + \log n)^{\mathcal{O}(1)}$.

Proof. We will prove the implications (i) $\Longrightarrow$ (iv) $\Longrightarrow$ (iii) $\Longrightarrow$ (ii) $\Longrightarrow$ (i)

(i) $\Longrightarrow$ (iv)
Assume $\mathrm{Kt}(x) = \mathrm{KDt}(x)^{O(1)}$ for every x. Let L be a set in P. In [BFL02] the authors show, that there is a polynomial time machine M with oracle access to L, such that for every $x \in L$ there is a description d_x of length $\log|L^{=|x|}| + O(\log n)$, such that the algorithm accepts (z, d_x) if and only if $z = x$. Since $L \in \mathsf{P}$, the machine M can answer the queries to L directly and thus for every $x \in L^{=|x|}$ we know $\mathrm{KDt}(x) = \log|L^{=|x|}| + O(\log n)$.

However, by assumption, KDt and Kt are polynomially related. Therefore $\mathrm{Kt}(x) = (|\log L^{=|x|}| + \log n)^{O(1)}$.

(iv) $\Longrightarrow$ (iii)

Let N be a nondetermistic machine running in time 2^n. Consider the following set $W_N \in \mathsf{P}$.

$$W_N = \left\{ w \;\middle|\; w \text{ is a witness for } x = \text{bin}|w| \text{ with respect to } N \right\} \quad .$$

By assumption, for every set L in P we have $\text{KT}_L(n) = (\log|L^{=n}| + \log n)^{\mathcal{O}(1)}$. In the following let n be given, such that $x = \text{bin}(n)$. Therefore $|x| = \log n$. If $N(x)$ has at most $2^{|x|^k}$ accepting computations, then $|W_N^{=n}| \leq 2^{|x|^k} = 2^{(\log n)^k}$. Thus, if $0 < |W_N^{=n}| \leq 2^{(\log n)^k}$ (i.e., x has few witnesses with respect to N), then there must be a witness w with $\text{KT}(w) \leq (\log|L^{=n}| + \log n)^{\mathcal{O}(1)} = (\log n)^{\mathcal{O}(1)} = |x|^c$ for some constant c.

The problem of finding a witness for x with respect to N if x has few witnesses can be solved by the following exponential time machine M. On input x the machine M generates all strings z with $\text{KT}(z) \leq (\log|x|)^c$. For each such string z the machine M checks if z is a witness for x with respect to N. If so, then M has established that $x \in L(N)$ and it outputs the string z. If none of the strings z are witnesses for x, either $x \notin L(N)$ or N has more than $2^{|x|^k}$ accepting computation paths. In either case, M outputs 0. Clearly M has the desired behavior as it will output a witness for x if N has few accepting paths on input x in exponential time.

It remains to be argued, that the machine M can be simulated by a family of polynomial size circuits. To see this, note that the assumption $\text{KT}_L(n) \leq (|\log L^{=n}| + \log n)^{\mathcal{O}(n)}$ for every $L \in \mathsf{P}$ implies that every P-printable set in $L \in \mathsf{P}$ must have $\text{KT}_L(n) \leq (\log n)^{\mathcal{O}(n)}$. (Note that every P-printable set contains at most $n^{\mathcal{O}(1)}$ string of length n.) Thus Theorem 4.6 yields $\mathsf{EXP} \in \mathsf{P}/_{\mathsf{poly}}$. We can conclude that there is a family of polynomial size circuits simulating M. That is, there is a circuit family $(C_i)_{i \in \mathbb{N}}$ such that $C_i(x, j) = j$th output bit of $M(x)$.

Therefore the search problem can be solved by polynomial size circuits.

Finally, a padding argument can be used to show that the ability to find witnesses for a nondeterministic computation running in time 2^n (provided that there are few such witnesses) using polynomial size circuits implies that FewEXP-search problems can be solved by polynomial size circuits.

(iii) $\Longrightarrow$ (ii)

Let N be nondeterministic machine running in time 2^{n^k} for some k. Note that by assumption there is family $(C_i)_{i\in\mathbb{N}}$ of circuits of size i^c, such that if x has at most $2^{|x|^k}$ witnesses with respect to N, then the string $w = C_i(x,1)C_i(x,2)\ldots C_i(x,2^{|x|^k})$ is a witness for x if N accepts x.

We argue that there is a deterministic machine M running in exponential time, such that if x has at most $2^{|x|^k}$ witnesses with respect to N, then the machine M accepts x if and only if $N(x)$ accepts. The machine M operates as follows. On input x the machine M computes all circuits C' of size i^c with $i = |x| + |x|^k$ input bits. For each such circuit M checks if $w' = C'(x,1)C'(x,2)\ldots C'(x,2^{|x|^k})$ is a witness for x with respect to N. If $N(x)$ has at most $2^{|x|^k}$ accepting paths and $x \in L(N)$, then one of the circuits C' will generate a witness for x. The machine M accepts if and only if one of the small circuits produces a witness for w. Clearly M runs in exponential time, and if $N(x)$ has few accepting paths, then M accepts the input x if and only if $x \in L(N)$. Thus M solves the FewEXP-decision problem for N in exponential time.

To see how we can obtain a small circuit family that solves the FewEXP-decision problem for N note the following. Consider the following nondeterministic machine N' that makes $2^{|x|^k}$ nondeterministic choices. On the computation path that always chooses 1, N' simulates M and accepts if and only if $M(x)$ accepts. On the computation path that always chooses 0, N' simulates M

and accepts if and only if $M(x)$ rejects. On all other computations paths N' always rejects. Note that N' has always exactly one accepting computation path. Thus every string x has always exactly one witness w_x of length $2^{|x|^k}$ with respect to N'. If M accepts x, then $w_x = 1^{2^{|x|^k}}$. If M rejects x, then $w_x = 0^{2^{|x|^k}}$. Thus computing the first bit of w_x suffices to determine the behavior of M. Computing the witness w_x for a given string x, is a FewEXP-search problem. By assumption there is a family of polynomial size circuits $(D_i)_{i\in\mathbb{N}}$ such that $w_x = D_i(x,1)D_i(x,2)\dots D_i(x,2^{|x|^k})$. Therefore, the circuit $D'_i(x) = D_i(x,1)$ accepts the input x if and only if $M(x)$ accepts and it consequently solves the FewEXP-decision problem for N.

(ii) $\Longrightarrow$ (i)

Let x be a string with $\mathrm{KDt}(x) = m$. That is, there is a description d_x, such that $U(d_x, z)$ accepts after $2^{m-|d|}$ steps, if and only if $z = x$.

Consider the NEXP machine N that on input $(d, 1^t, i, b, n)$ guesses a string $y \in \{0,1\}^n$, runs $U(d,y)$ for 2^t steps and then accepts if and only if $y_i = b$ and $U(d,y)$ accepts. If d_x is a distinguishing description for a string $x \in \{0,1\}^n$ and t is sufficiently large, then there is exactly one accepting path of N on input $(d_x, 1^t, i, x_i, |x|)$; there is no accepting path of N on $(d, 1^t, i, \overline{x_i}, |x|)$, for all $1 \le i \le |x|$. Since N has few accepting paths, our assumption yields that there is a family $(C_j)_{j\in\mathbb{N}}$ of polynomial size circuits, s.t. $C_j(d_x, 1^t, i, b, |x|)$ accepts, if and only if $x_i = b$. Given the description $d'_x = \langle C_j, d_x, t, |x|\rangle$, there is a trivial machine M' that accepts an input (d'_x, i, b) if and only if $x_i = b$ and runs in time $(m + \log n)^{\mathcal{O}(1)}$. The universal machine can simulate M' also in time $m^{\mathcal{O}(1)}$. Furthermore, C_j is a circuit of polynomial size, i.e. we have $|\langle C_j\rangle| = (m + \log n)^{\mathcal{O}(1)}$. Therefore $|d'_x| = (m + \log n)^{\mathcal{O}(1)}$ and thus $\mathrm{KT}(x) \le (m + \log n)^{\mathcal{O}(1)}$. □

Theorem 4.8. *The following statements are equivalent.*

(i) *For every string x,* $\mathrm{KT}(x) = (\mathrm{KNt}(x) + \log|x|)^{\mathcal{O}(1)}$.

(ii) $\mathsf{NEXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$.

(iii) NEXP*-search problems can be solved with polynomial-size circuits.*

(iv) *For every set* $L \in \mathsf{P}$, $\mathrm{KT}_L(n) = (\log n)^{\mathcal{O}(1)}$.

(v) *There is a dense set* $L \in \mathsf{P}/_{\mathsf{poly}}$, *such that* $\mathrm{KNt}_L(n) > n^{\varepsilon}$ *for some* ε.

Proof. We will prove the implications (ii) $\implies$ (i) $\implies$ (v) $\implies$ (ii) as well as the implications (ii) $\implies$ (iii) $\implies$ (iv) $\implies$ (ii)

(ii) $\implies$ (i)

Let x be a string with $\mathrm{KNt}(x) = m$ for some m. Thus there is a nondeterministic machine M and a description d of length m, such that $M(d, i, b)$ accepts if and only if $b = x_i$ in time 2^m. Consider the set $L_M = \{\langle d, i, b\rangle \mid M(d, i, b) \text{ accepts in } 2^{|d|} \text{ steps}\}$. Clearly $L_M \in \mathsf{NEXP}$ and by the assumption that $\mathsf{NEXP} \in \mathsf{P}/_{\mathsf{poly}}$, we can conclude that there is a circuit C of size m^c for some c, such that $C(d, i) = x_i$. If we hard-code d and $|x|$ into the circuit, we obtain a circuit C' of size $\mathcal{O}(m^c) + \log|x|$ computing f_x - the function with x as its truth-table. Now we can apply Theorem 3.15 and conclude that $\mathrm{KT}(x) \leq m^c + \log|x|^{\mathcal{O}(1)} = (\mathrm{KNt}(x) + \log|x|)^{\mathcal{O}(1)}$.

(i) $\implies$ (v)

Assume that $\mathrm{KT}(x) \leq (\mathrm{KNt}(x))^c$ for some constant c. This implies that $\mathrm{KT}(x) \leq (\mathrm{Kt}(x))^{\mathcal{O}(1)}$ and thus by Theorem 4.6 there is a dense set $L \in \mathsf{P}/_{\mathsf{poly}}$ with $\mathrm{Kt}_L(n) \geq n^{\varepsilon}$ for some $\varepsilon > 0$. However, the assumption $\mathrm{KT}(x) \leq (\mathrm{KNt}(x))^c$ also implies that $\mathrm{Kt}(x) \leq (\mathrm{KNt}(x))^{c'}$. Therefore, there is a dense set $L \in \mathsf{P}/_{\mathsf{poly}}$ with $\mathrm{KNt}_L(n) \geq n^{\frac{\varepsilon}{c'}}$.

(v) $\Longrightarrow$ (ii)
Assume that there is a dense set $L \in \mathsf{P}/\mathsf{poly}$ with $\mathrm{KNt}_L(n) \geq n^{\varepsilon}$ for some epsilon. By Theorem 5.36 we can conclude $\mathsf{NEXP}/\mathsf{poly} \subseteq \mathsf{P}^L/\mathsf{poly}$. As $L \in \mathsf{P}/\mathsf{poly}$ this implies $\mathsf{NEXP} \subseteq \mathsf{P}/\mathsf{poly}$.

(ii) $\Longrightarrow$ (iii)
This implication was shown in [IKW02].

(iii) $\Longrightarrow$ (iv)
Let $A \in \mathsf{P}$. Consider the following problem $L_A \in \mathsf{NE}$.

$$L_A = \left\{ x \;\middle|\; x = \mathrm{bin}(n) \text{ and } A^{=n} \neq \emptyset \right\} \quad .$$

Thus the associated NEXP-search problem is given by the machine M that accepts $\langle x, w \rangle$ if and only if $|w| = n$ for $n = \mathrm{int}(x)$ and $w \in L_A$. That is, the NEXP-search problem consists of finding a $w \in A$ of length of n. By assumption, there is a circuit of size $|x|^{\mathcal{O}(1)}$, such that $w = C(x,1)C(x,2)\ldots C(x,2^{|x|^c})$ is a witness for x. This means that if L_A contains strings of length $n = \mathrm{int}(x)$, then $w \in L_A$. Note that the circuit C provides a short description for w. The string w can be viewed as the truth-table of the function $f_x(i) = C(x,i)$. Since f_x can be computed by circuits of size $|x|^{\mathcal{O}(1)}$ we can use Theorem 3.15 and obtain $\mathrm{KT}(w) \leq |x|^{\mathcal{O}(1)} = (\log|w|)^{\mathcal{O}(1)}$.

(iv) $\Longrightarrow$ (ii)
Let $B \in \mathsf{NE}$. Let M_B be the deterministic verifier for B. Consider the following set $W_B \in \mathsf{P}$.

$$W_B = \left\{ w \;\middle|\; x = \mathrm{bin}(|w|) \text{ and } M_B(x,w) \text{ accepts} \right\} \quad .$$

By assumption every set in P contains strings with low KT-complexity for every length (for which it contains strings at all). Thus if $W_B^{=n} \neq \emptyset$ then there is a

w of length $|w| = n$ with $\mathrm{KT}(w) \leq (\log n)^{\mathcal{O}(1)}$. Thus if $x \in B$, there there is a witness w with $\mathrm{KT}(w) \leq (\log|x|)^c$ for some constant c.

The following deterministic machine M can decide B in time $2^{|x|^{\mathcal{O}(1)}}$. On input x the machine M generates all strings z with $\mathrm{KT}(z) \leq (\log|x|)^c$. For each such string M checks if the verifier $M_B(x, z)$ accepts. If so, then M established that $x \in B$. If not, then $M_B(x, z)$ will reject for every z of length $2^{|x|}$, as $W_B^{=n}$ is empty. Therefore $x \notin B$.

It remains to be argued, that the machine M can be simulated by a family of polynomial size circuits. Note that the assumption that for every $L \in \mathsf{P}$ we have $\mathrm{KT}_L(n) \leq (\log n)^{\mathcal{O}(n)}$ implies that every P-printable set in $L \in \mathsf{P}$ must have $\mathrm{KT}_L(n) \leq (\log n)^{\mathcal{O}(n)}$. Thus Theorem 4.6 yields that $\mathsf{EXP} \in \mathsf{P}/_{\mathsf{poly}}$. We can conclude that there is a family of polynomial size circuits simulating M and hence deciding $B \in \mathsf{NE}$. A padding argument yields that the conclusion $\mathsf{NE} \in \mathsf{P}/_{\mathsf{poly}}$ also implies $\mathsf{NEXP} \in \mathsf{P}/_{\mathsf{poly}}$. □

5 Complexity of Sets of Random Strings

In this section we discuss the complexity of sets containing only the strings with high resource-bounded Kolmogorov complexity.

5.1 Sets of Random Strings

In the resource-unbounded setting it is not difficult to determine the complexity of the set of Kolmogorov-random strings. For instance, let R_{K} denote the set that contains exactly all strings x with $\mathrm{K}(x) \geq \frac{|x|}{2}$. It is fairly easy to see that $R_{\mathrm{K}} \in \mathsf{co\text{-}RE}$, since we can enumerate all easy strings by dove-tailing over all descriptions. Also, it is not hard to see that $R_{\mathrm{K}} \notin \mathsf{REC}$ – otherwise the lexicographical first string x of each length in R_{K} would have a description of logarithmic length.

However, in the resource-bounded setting it is not quite as easy to establish the complexity of the sets of hard strings. Previous partial results in this direction have been obtained. In [BM95] Buhrman and Mayordomo showed that the set $\left\{x \mid \mathrm{K}_{\mathrm{t}}^{2^{n^2}}(x) \geq |x|\right\}$ is not complete for EXP under polynomial time Turing reductions. Buhrman and Torenvliet show in [BT01] that the set of strings with high conditional polynomial space-bounded Kolmogorov complexity (namely sets $\left\{x \mid \mathrm{K}_{\mathrm{s}}^{p(n)}(x|y) \geq |x|\right\}$, for some polynomial p,) is hard for PSPACE under non-deterministic polynomial time Turing reductions. The results in this section give significantly stronger observations about sets of K-random strings for Kt-style complexity measures.

Definition 5.1. *Let* Kμ *be an arbitrary Kolmogorov complexity measure. Define* $R_{\mathrm{K}\mu}$*, the set of* Kμ*-random strings, as*

$$R_{\mathrm{K}\mu} = \left\{x \mid \mathrm{K}\mu(x) \geq \frac{|x|}{2}\right\} \quad .$$

Note that the choice of the randomness threshold $\frac{|x|}{2}$ *is arbitrary. Any value between* $|x|^{\varepsilon}$*, for* $0 < \varepsilon \leq 1$*, and* $|x|$ *would suffice for the theorems discussed in this thesis.*

It is interesting to note that even though the set $R_{\mathrm{K}\mu}$ as defined above is very similar to a set, say, $R'_{\mathrm{K}\mu} = \left\{x \;\middle|\; \mathrm{K}\mu(x) \geq \frac{|x|}{3}\right\}$, there does not seem to be a clear direct reduction between these two sets. If we consider for instance the sets R_{Kt} and R'_{Kt}, we will see that both are complete for EXP under $\mathsf{P}/_{\mathsf{poly}}$ reductions, and thus they are reducible to one another by a family of polynomial size circuits. But there does not seem to be a way to construct a direct (say Turing)-reduction from R_{Kt} to R'_{Kt}. In the remainder of this section we will prove completeness results for different measures $\mathrm{K}\mu$ for different complexity classes. These results are somewhat surprising, since the sets $R_{\mathrm{K}\mu}$ do not seem to posses an inherent combinatorial structure that is usually characteristic for complete problems. Thus it is quite surprising that we can show that it is possible to gain computational power from these sets through explicit reductions. The advances in the area of derandomization are the basis for the completeness results discussed in this section.

5.2 Tools from Derandomization

Our main tool to establish our results will be pseudo random generators (PRGs). A pseudo random generator is a program that, based on a short random seed, produces a long output which seems random to resource-bounded (say, polynomial time bounded) algorithms. This notion initially goes back to Blum and Micali, and Yao [BM84, Yao82].

Definition 5.2 (Pseudo Random Generator). *A* pseudo random generator (PRG)*, is a family of functions* $G = (G_n : \{0,1\}^{s(n)} \mapsto \{0,1\}^n)_{n\in\mathbb{N}}$ *that takes as input a seed of* $s(n)$ *bits and produces as output a pseudo random string with* n *bits. In a slight abuse of notation, we also denote the entire generator as*

$G : \{0,1\}^{s(n)} \mapsto \{0,1\}^n$.

In order to determine if a given generator can be used to derandomize a probabilistic algorithm, we need to know if that algorithm behaves essentially the same way, when given pseudo random bits instead of true randomness.

Definition 5.3 (Distinguishing random and pseudo random). *If G is a pseudo random generator and A is a set, then A* distinguishes the output of G from truly random *bits, if there is a polynomial p, such that*

$$|\underset{r\in U_n}{\mathbf{Pr}}\big[\, r \in A \,\big] - \underset{s\in U_{s(n)}}{\mathbf{Pr}}\big[\, G(s) \in A \,\big]| > \frac{1}{p(n)}$$

If we want to know if a given generator G is sufficiently strong to derandomize a randomized algorithm M_R, we need to determine if there are infinitely many inputs x on which M_R behaves significantly differently when given pseudo random strings instead of truly random ones.

Definition 5.4 (Successful Derandomization). *A generator G successfully derandomizes a probabilistic algorithm M_R, if for all but finitely many x, none of the sets $A_x = \{r \mid M_R(x)$ computes the correct output using random bits $r\}$ can distinguish random and pseudo random strings.*

The security of a generator is usually measured against a complexity class.

Definition 5.5 (Secure PRG**).** *A pseudo random generator G is* secure against a class C*, if no set $A \in \mathsf{C}$ can distinguish the output of G from truly random bits.*

The definitions imply for instance that if a generator is secure against $\mathsf{P}/_{\mathsf{poly}}$, then every set in BPP can be derandomized.

Most of the generators that are relevant to this thesis are based on the approach introduced in Nisan and Wigderson's seminal paper [NW94]. In their paper, they

construct a generator G_f^{NW} based on a function $f \in \mathsf{EXP}$ that cannot be approximated by polynomial size circuits. This generator has the property, that for infinitely many input lengths, a BPP-algorithm cannot distinguish the pseudo-random strings from truly random strings. Thus the algorithm can be derandomized on these input lengths. Babai, Fortnow, Nisan, and Wigderson showed in [BFNW93] that a weaker assumption suffices. They construct a NW-style generator G_f^{BFNW} based on a function, that is not *computable* by polynomial size circuits. Klivans and van Melkebeek observed [KvM02] that the construction of G_f^{BFNW} also holds relative to an oracle. We state this result in the contrapositive.

Theorem 5.6 ([BFNW93, KvM02]). *Let f be a Boolean function, $\varepsilon > 0$, and $G_f^{BFNW} : \{0,1\}^{n^\varepsilon} \mapsto \{0,1\}^n$ be the pseudorandom generator described above. Let T be a set and $p(n)$ a polynomial. If for all large n*

$$|\Pr_{r \in U_n}[r \in T] - \Pr_{x \in U_{n^\varepsilon}}[G_f^{BFNW}(x) \in T]| \geq \frac{1}{p(n)} \quad ,$$

then there exists a family $(C_n)_{n \in \mathbb{N}}$ of circuits of polynomial size with oracle T that computes f and queries T non-adaptively.

In order to apply this theorem to get hardness results for $R_{\mathrm{K}\mu}$, it is important how quickly the output $G_f^{BFNW}(s)$ can be computed for a given seed s. The following terminology will help us to state a fairly low upper bound on the time required for this computation.

Definition 5.7 (PSPACE-robust). *A set A is* PSPACE-robust*, if $\mathsf{P}^A = \mathsf{PSPACE}^A$.*

Recall that a computation running in PSPACE^A can only make queries to the oracle of length at most $n^{\mathcal{O}(1)}$ by definition.

Further note that complete sets for most sufficiently large complexity classes (such as PSPACE, EXP, $\mathsf{EXPSPACE}$, ...) are PSPACE-robust.

The analysis of the generator construction in [BFNW93] yields the following.

Fact 5.8. *Let $G_f^{BFNW} : \{0,1\}^{n^\varepsilon} \mapsto \{0,1\}^n$ be the generator discussed above, stretching n^ε bits to n truly random bits. For every seed $s \in \{0,1\}^{n^\varepsilon}$, the value $G_f^{BFNW}(s)$ is computable in space $\mathcal{O}(n^\varepsilon)$ given oracle access to f. The oracle f is only queried for inputs q of length $|q| = \mathcal{O}(n^\varepsilon)$. If f is* PSPACE*-robust, then each bit of $G_f^{BFNW}(s)$ can be computed in time $\mathcal{O}(n^{c\varepsilon})$ for some constant c independent of ε, given oracle access to f.*

In [IW98], Impagliazzo and Wigderson reexamine the approach of [BFNW93]. In the previous constructions, in order for the generator to be secure, the function f on which the generator G_f^{BFNW} was based needed to be non-uniformly hard. That is, it was required that the function could not be computed by small circuits. Impagliazzo and Wigderson showed that it is possible to construct a similar generator G_f^{IW98} that is based on a function f that is uniformly hard. That is, the function f cannot be efficiently computed by a (possibly probabilistic) Turing machine. This construction will allow us to derive uniform hardness results for R_{Kt} and R_{KS}. Again, we state the main result of [IW98] in the contrapositive - the way we will apply it. For definition of *random self-reducibility* and *downward self-reducibility*, refer to Section 2.

Theorem 5.9 ([IW98]). *Let $f : \{0,1\}^* \mapsto \{0,1\}^*$ be a random self-reducible and downward self-reducible function. Let $\varepsilon > 0$. Then there is a pseudorandom generator $G_f^{IW98} : \{0,1\}^{n^\varepsilon} \mapsto \{0,1\}^n$ with the following properties.*
The output of $G_f^{IW98}(s)$ is computable in space $\mathcal{O}(n^\varepsilon)$, given oracle access to f. Furthermore, if T is a set and $p(n)$ is a polynomial, and if for all large n

$$|\Pr_{r \in U_n}[r \in T] - \Pr_{x \in U_{n^\varepsilon}}[G_f^{IW98}(x) \in T]| > \frac{1}{p(n)} \quad ,$$

then there exists a probabilistic Turing machine, with oracle T, running in polynomial time that, on input x, outputs $f(x)$ with probability at least $\frac{2}{3}$.

The preceding two theorems provide the key derandomization techniques that are required to prove our completeness results. They are stated in the contrapositive of their original formulations, since that is the way we will use them. However, some of our completeness results (namely for EXP and PSPACE) involve uniform reductions that make use of randomness. These results can be further strengthened by derandomizing the reductions using, again, pseudo random generators - this time in the "traditional" way. We will make use of the generator G_f^{IW97} presented in [IW97] (see also [STV01]). In this context, the generator stretches the seed exponentially and is thus suitable to completely derandomize polynomial time computation. Again, Klivans and van Melkebeek observe that the construction relativizes.

Theorem 5.10 ([IW97, KvM02]). *For any $\varepsilon > 0$, there exist constants $c, c' > 0$ such that the following holds. Let A be a set and $n > 1$ be an integer. Let $f : \{0,1\}^{c \log n} \mapsto \{0,1\}$ be a Boolean function that cannot be computed by oracle circuits of size $n^{c\varepsilon}$ with oracle A. Then $G_f^{IW97} : \{0,1\}^{c' \log n} \mapsto \{0,1\}^n$ satisfies:*

$$|\Pr_{r \in U_n}[C^A(r) = 1] - \Pr_{x \in U_{c' \log n}}[C^A(G_f^{IW97}(x)) = 1]| < \frac{1}{n} \quad ,$$

for any oracle circuit C^A of size at most n.

For x of size $c' \log n$, $G_f^{IW97}(x)$ is computable in time polynomial in n given access to f on inputs of length $c \log n$.

We will also need to apply hardness versus randomness tradeoffs for pseudorandom generators G_f of lower computational complexity than the Nisan-Wigderson-style generators considered above, in order to prove hardness results for R_{KT}. We will use the cryptographic pseudorandom generators that developed out of the seminal work by Blum and Micali [BM84], and by Yao [Yao82]. In [HILL99], it is shown how to construct such a pseudorandom generator $G_f^{HILL} : \{0,1\}^n \mapsto \{0,1\}^{2n}$ out of any function f. The pseudorandom generator G_f^{HILL} is computable in polynomial time

given access to f, and is secure provided f is one-way.

We will apply G_f^{HILL} for functions f that take some additional parameters y besides the actual input x. In the case of the Discrete Log problem, for example, y will describe the prime p and the generator g of $\mathbb{Z}_p^*$ defining the basis for the logarithm. More precisely, we will consider functions of the form $f(y,x)$ that are length-preserving for every fixed y, i.e., $f_y : \{0,1\}^n \mapsto \{0,1\}^n$, where $f_y(x) = f(y,x)$. The function $f(y,x)$ will be computable uniformly in time polynomial in $|x|$ (so without loss of generality, $|y|$ is polynomially bounded in $|x|$). The hardness versus randomness tradeoff in [HILL99] can be stated as follows.

Theorem 5.11 ([HILL99]). *Let $f(y,x)$ be computable uniformly in time polynomial in $|x|$. For any oracle L, polynomial-time probabilistic oracle Turing machine M, and polynomial p, there exists a polynomial-time probabilistic oracle Turing machine N and polynomial q such that the following holds for any n and y: If*

$$|\mathop{\mathbf{Pr}}_{r\in U_{2n},s}\big[\,M^L(y,r,s)=1\,\big]-\mathop{\mathbf{Pr}}_{x\in U_n,s}\big[\,M^L(y,G_{f_y}^{HILL}(x),s)=1\,\big]|>\frac{1}{p(n)}$$

then

$$\mathop{\mathbf{Pr}}_{x\in U_n,s}\big[\,f(y,N^L(y,f(y,x),s))=f(y,x)\,\big]\geq\frac{1}{q(n)}$$

where s denotes the internal coin flips of M or N, respectively.

5.3 "Inverting Pseudo Random Generators"

All results about completeness / hardness of a set $R_{\mathrm{K}\mu}$ in this section are based on the same idea. As mentioned in Section 3, the measure Kμ can be interpreted as a measure indicating the "randomness" of a string. Thus, a set $R_{\mathrm{K}\mu}$ can be viewed as the problem of distinguishing random from non-random strings. The ability to tell

random and pseudo-random strings apart is a crucial factor in the question of security of pseudo random generators. The correctness proofs of all NW-style generators proceed along the same line. The ability of an algorithm to distinguish random from pseudo random strings produced by a generator G_f gives rise to a circuit computing the hard function f on which the generator is based. From an intuitive perspective, it should not be surprising that indeed a set $R_{\mathrm{K}\mu}$ can serve as an oracle to compute the hard function f.

Let us state this idea more precisely in Theorem 5.12. Note that most Kt-style complexity measures are linearly related to KT^A for an appropriately chosen set A. Thus, with an appropriate choice of γ the following theorem yields hardness results for $R_{\mathrm{K}\mu}$ for every Kt-style measure $\mathrm{K}\mu \geq \mathrm{KS}$.

Theorem 5.12. *Let A be any* PSPACE*-robust set. Let L be a set of polynomial density such that for every $x \in L$, $\mathrm{KT}^A(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$. Then A is reducible to L via $\leq_{\mathrm{tt}}^{\mathsf{P/poly}}$ reductions.*

Proof. Let A be a PSPACE-robust set and let f denote the characteristic function of A. Let $0 < \gamma < 1$ be fixed. Consider the generator $G_f^{BFNW} : \{0,1\}^{n^\varepsilon} \mapsto \{0,1\}^n$, where the choice of ε is determined as follows. We know that, given access to A, every bit of G_f^{BFNW} is computable in time $n^{\varepsilon c}$ for some constant c independent of ε. We can assume that $c > 1$ and set $\varepsilon = \frac{\gamma}{2}$.

We claim that any string in the range of G_f^{BFNW} has small KT^A complexity. Let M^A be the machine computing $G_f^{BFNW}(s)$, in time $n^{\frac{\gamma}{2}}$ with access to the oracle A, for any $s \in \{0,1\}^{n^\varepsilon}$. Thus the description $d = \langle M, s\rangle$ suffices, such that the universal machine with oracle A accepts the input $\langle d, i, b\rangle$ if and only if the ith bit of x is b. This computation takes at most $n^{\frac{1}{2}\gamma}\log(n^{\frac{1}{2}\gamma}) \leq \mathcal{O}(n^{\frac{2}{3}\gamma})$ steps. This gives the following upper bound: $\mathrm{KT}^A(y) \leq |d| + \mathcal{O}(n^{\frac{2}{3}\gamma}) = \mathcal{O}(n^\varepsilon) + \mathcal{O}(n^{\frac{2}{3}\gamma}) = \mathcal{O}(n^{\frac{2}{3}\gamma})$. Since L contains only strings with $\mathrm{KT}^A \geq n^\gamma$, it follows for sufficiently large n that

$\mathop{\mathbf{Pr}}_{s\in U_{n^\varepsilon}}\big[\, G_f^{BFNW}(s) \in L \,\big] = 0$. On the other hand, L is of polynomial density, i.e. there is a polynomial p, such that $\mathop{\mathbf{Pr}}_{r\in U_n}\big[\, r \in L \,\big] \geq \frac{1}{p(n)}$. Thus for all large n,

$$|\mathop{\mathbf{Pr}}_{r\in U_n}\big[\, r \in L \,\big] - \mathop{\mathbf{Pr}}_{s\in U_{n^\varepsilon}}\big[\, G_f^{BFNW}(s) \in L \,\big]| \geq \frac{1}{p(n)} \quad .$$

Theorem 5.6 yields that there is a circuit family $(C_n)_{n\in\mathbb{N}}$ of polynomial size that queries L non-adaptively and computes f. Thus, the function f is $\leq_{\mathrm{tt}}^{\mathsf{P}/\mathsf{poly}}$ reducible to L. □

5.4 Complexity of R_{Kt}

In this subsection we will discuss the complexity of the set R_{Kt}. It is fairly straightforward to provide a natural upper bound for R_{Kt}.

Theorem 5.13. $R_{\mathrm{Kt}} \in \mathsf{E}$.

Proof. In order to determine if a string $x \in R_{\mathrm{Kt}}$, a Turing machine M needs to enumerate at most $2^{\frac{|x|}{2}+1} - 1$ descriptions d of length $\frac{|x|}{2}$. For each such description M needs to simulate the universal machine for at most $2^{\frac{|x|}{2}}$ steps. Thus the run-time of M is bounded from above by $\mathcal{O}(2^n)$. □

At present, the best lower bound that we can provide for R_{Kt} is the following result from [All01].

Fact 5.14. $R_{\mathrm{Kt}} \notin \mathsf{AC}^0$.

This fact is quite unsatisfactory. It leaves a huge gap between the upper bound and the lower bound for the complexity of R_{Kt}. Furthermore, we will see below that R_{Kt} is actually complete for EXP under reductions computable by polynomial size circuits, and this suggests that possibly $R_{\mathrm{Kt}} \notin \mathsf{P}/\mathsf{poly}$.

Currently it does not seem feasible to prove this fact (as this would settle the important open question whether EXP can be decided by polynomial size circuits), but

at least one might hope for a lower bound of $R_{\mathrm{Kt}} \notin \mathsf{P}$. However, even for this case we are only able to provide a conditional result. The following corollary follows from Theorem 5.17 and Theorem 5.26.

Corollary 5.15. $R_{\mathrm{Kt}} \notin \mathsf{ZPP}$, *unless* $\mathsf{EXP} = \mathsf{ZPP}$.

Proof. Theorem 5.17 yields $\mathsf{EXP} \subseteq \mathsf{NP}^{R_{\mathrm{Kt}}}$. The proof of Theorem 5.26 also yields $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{R_{\mathrm{Kt}}}$. The assumption $R_{\mathrm{Kt}} \in \mathsf{ZPP}$ simplifies these two statements to $\mathsf{EXP} = \mathsf{NP}^{\mathsf{ZPP}} \subseteq \mathsf{PSPACE}$ and $\mathsf{PSPACE} = \mathsf{ZPP}^{\mathsf{ZPP}} = \mathsf{ZPP}$. Thus $\mathsf{EXP} = \mathsf{ZPP}$. □

We can characterize the complexity of R_{Kt} much more precisely, by considering its completeness properties. A direct application of Theorem 5.12 for Kt yields the following result.

Theorem 5.16. R_{Kt} *is complete for* EXP *under* $\leq_{\mathrm{tt}}^{\mathsf{P/poly}}$ *reductions.*

Proof. Let A be a complete set for E under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reductions. It is straightforward to verify, that A is PSPACE-robust. Note that $\mathrm{cens}_{R_{\mathrm{Kt}}}(n) \geq 2^n - 2^{\frac{n}{2}}$. Thus R_{Kt} has more than polynomial density. Next, recall (→Theorem 3.20) that there is a constant $c > 0$, such that $c \cdot \mathrm{Kt}(x) > \mathrm{KT}^A$. Therefore, for every $x \in R_{\mathrm{Kt}}$, it holds that $\mathrm{KT}^A(x) \geq \frac{1}{2c}|x|$. Thus any $\varepsilon < 1$ yields $\mathrm{KT}^A(x) \geq |x|^\varepsilon$ for all sufficiently long x. Now Theorem 5.12 directly gives the desired result. □

This hardness result does not only apply to R_{Kt}, but to any dense set containing no strings of low resource-bounded Kolmogorov complexity (including the Buhrman-Mayordomo set $R_{\mathrm{K}_{\mathrm{t}}^{2^{n^2}}}$ and the set R_{K} of high, unbounded K-complexity).
The statement of Theorem 5.16 can be strengthened to show completeness under nondeterministic, but uniform reductions.

Theorem 5.17. R_{Kt} *is complete for* EXP *under* $\leq_{\mathrm{T}}^{\mathsf{NP}}$ *reductions.*

In order to prove this theorem, we will argue that $\mathsf{EXP} \subseteq \mathsf{MA}^{R_{\mathrm{Kt}}}$. We will then use the fact that the set R_{Kt} suffices as oracle to derandomize MA.

Lemma 5.18. *Let A be any oracle and L be a set such that $L \in \mathsf{P}^A/_{\mathsf{poly}}$ and for every $x \in L$, $\mathrm{KT}^A(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$. If L contains a string of every length, then $\mathsf{MA}^L \subseteq \mathsf{NP}^L$.*

Proof. The NP-machine M guesses a string $\chi_f \in L$ of length m, for $m = n^c$, and interprets it as the truth-table of a Boolean function f on inputs of size $\lfloor \log m \rfloor$. Since $\chi_f \in L$, we have $\mathrm{KT}^A(\chi_f) \geq m^\gamma$. Thus by Theorem 3.15, f requires circuits of size at least $\Omega(m^{\gamma/2})$ even when the circuit has access to the oracle A. We need to establish that f requires large circuits even when these have access to the oracle L instead of A. To see this, note the following. By assumption, L is reducible to A by circuits of size m^k for some k. Thus, if C is an L-oracle circuit of size s then there is an A-oracle circuit of size $s \cdot s^k = s^{k+1}$ computing the same function as C. If f requires A oracle circuits of size $\Omega(m^{\gamma/2})$, then f requires L oracle circuits of size at least $(\Omega(m^{\gamma/2}))^{\frac{1}{k+1}} \geq \Omega(m^{\gamma/4k})$.

The function f is of sufficient hardness to construct a generator G_f^{IW97}, as stated in Theorem 5.10, to stretch $\mathcal{O}(\log m)$ random bits into m pseudorandom bits. The output of this generator is indistinguishable from random by any oracle circuit of size n with access to L. Thus we can fully derandomize the probabilistic part of the MA-computation. □

Applying the previous lemma with an oracle A that is complete for E under linear-time reductions yields the following corollary.

Corollary 5.19. $\mathsf{MA}^{R_{\mathrm{Kt}}} = \mathsf{NP}^{R_{\mathrm{Kt}}}$.

Proof of Theorem 5.17. The inclusion $\mathsf{NP}^{R_{\mathrm{Kt}}} \subseteq \mathsf{EXP}$ is trivial, so we focus on the other inclusion. A consequence of Theorem 5.16 is that $\mathsf{EXP} \subseteq \mathsf{MA}^{R_{\mathrm{Kt}}}$. To see this, let A be any language in EXP. The set A can be decided by a 2-prover interactive proof system with provers computable in EXP [BFL91]. By Theorem 5.16, these provers can be implemented in two polynomial size oracle circuits with access to oracle R_{Kt}. The

$\mathsf{MA}^{R_{\mathrm{Kt}}}$ protocol is as follows. Merlin sends Arthur these two circuits that compute the answers given by the two provers for A when given access to R_{Kt}. Arthur then executes the MIP protocol, simulating the provers' answers by executing the circuits and querying the oracle R_{Kt}. The theorem now follows from Corollary 5.19 □

In a sense the completeness result in Theorem 5.16 is optimal. We will see that for every polynomial p, there is a set $L \in \mathsf{EXP}$ that is not truth-table reducible to R_{Kt} in time $2^{p(n)}$ given $p(n)$ bits of advice.

Notation 5.20. Define the class C_{tt}^k as

$$\mathsf{C}_{tt}^k = \left\{ L \;\middle|\; L \text{ is truth-table reducible to } R_{\mathrm{Kt}} \text{ in time } 2^{n^k} \text{ with } n^k \text{ bits of advice} \right\} .$$

The following theorem states, that for every k there is a set in EXP, that cannot be reduced to R_{Kt} in time 2^{n^k}, even when given n^k bits of advice.

Theorem 5.21. *For every k, there is a set $L \in \mathsf{EXP}$, such that $L \notin \mathsf{C}_{tt}^k$.*

This theorem is a direct corollary from the following measure-theoretic statement. For the necessary background on resource-bounded measure, refer to Section 2.

Theorem 5.22. *For every $k > 0$, the measure $\mu(\mathsf{C}_{tt}^k|\mathsf{EXP}) = 0$.*

The key observation is the following Lemma. Truth table reductions to R_{Kt} only need to ask queries up to a certain length. This fact was initially observed for $\mathrm{K}_{\mathrm{t}}^{2^{n^2}}$ in [BM95].

Lemma 5.23. *Let $k > 0$ be fixed. Assume $L \in \mathsf{C}_{tt}^k$ via oracle machine M_1 and advice function a, such that on input $\langle x, a(|x|)\rangle$ the machine M_1 runs in time $t(n) = 2^{n^k}$ for $n = |x|$. There is a machine M_2 with oracle R_{Kt} and advice function a, that decides L in time $\mathcal{O}(2^{n^{k+1}})$ and that only asks queries q of length $|q| \leq 2|x|^{k+1}$.*

Proof. Let L be truth-table reducible to R_{Kt} via machine M_1 in time 2^{n^k} with advice a of length n^k. Let x be a sufficiently long string, and let $q_1, \ldots, q_m$ be the queries asked by M_1 on input $\langle x, a(|x|)\rangle$. Let $n = |x|$.

The main idea to prove this lemma is the fact that the queries asked have a reasonably short description depending on x and $a(n)$, but not on the length of the query. Thus, any possible long query is fairly simple when measured relative to its length.

Assume that $|q_i| > 2n^{k+1}$. We argue that $q_i \notin R_{\mathrm{Kt}}$. We want to argue that there is a short description d_{q_i} such that the universal machine accepts the input $\langle d_{q_i}, j, b\rangle$ if and only if the j-th bit of q_i is b. Note that, given x, advice $a(n)$, i, index j, and bit b, a machine M' can simulate M_1 until the query tuple $(q_1, \ldots, q_m)$ is generated. Then M' accepts, if and only if the j-th bit of q_i is b. This simulation takes $\mathcal{O}(2^{n^k})$ steps. The universal machine can simulate M' given the description $d = \langle M', \langle x, a(n), i\rangle\rangle$ in time $\mathcal{O}(2^{n^k \log n^k})$. Since $|\langle M', \langle x, a(n), i\rangle\rangle| = \mathcal{O}(n^k)$, the Kt complexity of q_i can be bounded by $\mathrm{Kt}(q_i) \leq |d| + \log(\mathcal{O}(2^{n^k})) = \mathcal{O}(n^k) < n^{k+1}$. Since $|q_i| > 2n^{k+1}$, we have $\mathrm{Kt}(q_i) < \frac{1}{2}|q_i|$ and therefore $q_i \notin R_{\mathrm{Kt}}$.

Therefore L can be reduced to R_{Kt} by a machine M_2, that simulates M_1, but answers all queries q to R_{Kt} of length $|q| > 2|x|^{k+1}$ directly as No. □

Now we can use Lemma 5.23 to prove Theorem 5.22.

Proof of Theorem 5.22. Let $k \geq 1$ be fixed. Let $(M_j)_{j\in\mathbb{N}}$ be an enumeration of oracle Turing machines running in time 2^{n^k}, such that each M_i appears in this sequence infinitely often. Assume that each M_i queries the oracle non-adaptively and on input $(x, a(|x|))$, the machine M_i never asks a query q of length $|q| > n^{k+1}$. For each machine M_i we define a set X_i that contains all languages that can be decided by M_i with some advice function of length at most n^k. More formally, define X_i as

follows:

$$X_i = \left\{ L \;\middle|\; \forall n \;\exists\, a \in \{0,1\}^{n^k} \;\forall x : M_i^{R_{\mathrm{Kt}}}(x,a) = 1 \text{ iff } x \in L \right\} \quad .$$

It follows from Lemma 5.23 that $\mathsf{C}^k_{tt} \subseteq \bigcup_i X_i$. We will provide a uniform family $(d_i)_{i\in\mathbb{N}}$ of martingales where every d_i succeeds on X_i. Therefore every $L \in \mathsf{C}^k_{tt}$ will be covered by some martingale d_i, and thus by Lemma 2.13 this suffices to show $\mu(\mathsf{C}^k_{tt}|\mathsf{EXP}) = 0$.

The family $(d_i)_{i\in\mathbb{N}}$ is defined as follows. For every i, we will define the function d_i on strings $w \in \{0,1\}^*$ which we view as prefixes of infinite characteristic sequences of languages. Let $n = \lfloor \log|w| + 1 \rfloor$. We interpret $w = t_0 t_1 \dots t_{n-1} t_n$ as a concatenation of truth-tables, where each table t_j (for $j < n$) is of length 2^j.

Let $x_1, \dots, x_{2^n}$ denote all of the strings of length n. That is, t_n can be viewed as (part of) the truth table of some characteristic function on $x_1, \dots, x_{2^n}$.

Let $\#\mathrm{Adv}(w)$ denote the number of advice strings for which the computation of $M_i^{R_{\mathrm{Kt}}}$ agrees with t_n. If $t_n[j]$ denotes the jth bit of t_n, then

$$\#\mathrm{Adv}(w) = \left|\left\{ a \in \{0,1\}^{n^k} \;\middle|\; M_i^{R_{\mathrm{Kt}}}(x_j, a) = t_n[j] \text{ for all } 1 \le j \le |t_n| \right\}\right| \quad .$$

Now we can define the martingale d_i. Given $w \in \{0,1\}^*$ and $b \in \{0,1\}$, let $n = \lfloor \log|wb| + 1 \rfloor$. Note $\frac{\#\mathrm{Adv}(wb)}{\#\mathrm{Adv}(w)}$ is the fraction of those advice strings for which the computation of $M_i^{R_{\mathrm{Kt}}}$ agrees with w, where the computation of $M_i^{R_{\mathrm{Kt}}}$ also agrees with wb. Define

$$d_i(wb) = \begin{cases} 1 & \text{for } w = \lambda \\ d_i(w) \cdot 2 \cdot \frac{\#\mathrm{Adv}(wb)}{\#\mathrm{Adv}(w)} & \text{otherwise} \end{cases}$$

It is straightforward to see that the function d_i satisfies the condition $d_i(w) =$

$\frac{1}{2}(d_i(w0) + d_i(w1))$ and is thus a martingale.

Let us verify that $d_i(wb)$ can be computed in quasi-polynomial time in terms of $|wb|$. We can compute $\#\mathrm{Adv}(wb)$ by cycling through all possible advice strings a of length $|a| = n^k$ and simulating $M_i^{R_{\mathrm{Kt}}}(x_j, a)$ for all $1 \leq j \leq |t_n|$ to verify that a agrees with t_n. Since M_i does not ask queries longer than n^{k+1}, any query q to R can be answered in time $2^{n^{k+1}} \cdot 2^{(n^{k+1})^2} < 2^{n^{3k}}$. As the run-time of M_i is bounded by 2^{n^k}, the computation of $\#\mathrm{Adv}(wb)$ takes at most $2^{n^k} + 2^{n^k} \cdot 2^{n^{3k}} = 2^{\mathcal{O}(n^{3k})} = 2^{\mathcal{O}((\log|w|)^{3k})}$. Thus the value $d_i(wb)$ can be computed in quasi-polynomial time.

It remains to be shown that given $L \in X_i$, the martingale d_i succeeds on χ_L. View the infinite binary sequence $\chi_L = t_1 t_2 t_3 \ldots$ as a concatenation of the truth-tables $t_1, t_2, \ldots$ with $|t_j| = 2^j$. We will show that for all sufficiently large j, the value $d_i(t_1 \ldots t_j) > 2 \cdot d_i(t_1 \ldots t_{j-1})$. Denote $w = t_1 t_2 \ldots t_{j-1}$. Note that

$$\begin{aligned} d_i(wt_j) &= d_i(w) \prod_{l=1}^{2^n} 2 \cdot \frac{\#\mathrm{Adv}(wt_j[1..l])}{\#\mathrm{Adv}(wt_j[1..l-1])} \\ &= d_i(w) \cdot 2^{2^n} \cdot \frac{\#\mathrm{Adv}(wt_j)}{\#\mathrm{Adv}(w)} \\ &\geq d_i(w) \cdot 2^{2^n} \cdot \frac{1}{2^{n^k}} = d_i(w) \cdot 2^{2^n - n^k} \quad . \end{aligned}$$

The last inequality follows from the fact that there is an upper bound of 2^{n^k} on $\#\mathrm{Adv}(w)$, as there are only that many possible advice strings. Furthermore, there must be an advice string that agrees with t_j, since $w = \chi_L$ and $L \in X_i$. Thus $\#\mathrm{Adv}(wt_j) \geq 1$.

Note that even for small j there is always at least one advice string for which the computation $M_i^{R_{\mathrm{Kt}}}$ agrees with t_j. Therefore, even for small j the value of the martingale remains positive. This guarantees that $d_i(w)$ grows unbounded as w grows to longer and longer prefixes of χ_L. □

5.5 Completeness results for KS

The results in this subsections are along similar lines as in Subsection 5.4.

Theorem 5.24. $R_{\mathrm{KS}} \in \mathsf{PSPACE}$

Proof. The proof is almost identical to the proof of Theorem 5.13 □

As in the case for R_{Kt}, we are not able to provide meaningful lower bounds on the complexity of R_{Kt}. From [All01] we get that $R_{\mathrm{KS}} \notin \mathsf{AC}^0$. One might expect to have a lower bound of $R_{\mathrm{KS}} \notin L$. However, at this point we are not able to unconditionally prove such a result. Our best tool to pinpoint the complexity of R_{KS} is again to show completeness and non-completeness results under different reductions.

Theorem 5.25. *R_{KS} is complete for PSPACE under $\leq_{\mathrm{tt}}^{\mathsf{P/poly}}$ reductions.*

Proof. The proof is essentially identical to the proof of Theorem 5.16. Let A be a complete set for $\mathsf{DSpace}[n]$ under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reductions. It is straightforward to verify that A is PSPACE-robust. Note that $\mathrm{cens}_{R_{\mathrm{KS}}}(n) \geq 2^n - 2^{\frac{n}{2}}$. Thus R_{KS} has more than polynomial density. Next, recall (→Theorem 3.20) that there is a constant $c > 0$, such that $c \cdot \mathrm{KS}(x) > \mathrm{KT}^A(x)$. Therefore, for every $x \in R_{\mathrm{KS}}$, it holds that $\mathrm{KT}^A(x) \geq (\frac{1}{2c}|x|)$. Thus any $\varepsilon < 1$ yields $\mathrm{KT}^A(x) \geq |x|^\varepsilon$ for all sufficiently long x. Now Theorem 5.12 directly gives the desired result. □

The statement of Theorem 5.25 can be strengthened to show completeness under randomized, but uniform reductions.

Theorem 5.26. *R_{KS} is complete for PSPACE under $\leq_{\mathrm{T}}^{\mathsf{ZPP}}$ reductions.*

In order to prove this theorem, we will argue that $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{R_{\mathrm{KS}}}$. We will then use that the set R_{KS} suffices as oracle to derandomize BPP to a zero-error randomized algorithm. Further, we will also show that R_{K} is sufficient as an oracle to completely derandomize derandomize the BPP computation and thus place $\mathsf{PSPACE} \subseteq \mathsf{P}^{R_{\mathrm{K}}}$.

Lemma 5.27. *Let A be any oracle and L be a set such that $L \in \mathsf{P}^A/_{\mathsf{poly}}$ and every $x \in L$ has $\mathrm{KT}^A(x) > |x|^\gamma$, for some constant $0 < \gamma < 1$. If there is a polynomial $p(n)$, such that $\mathrm{cens}_L(n) \geq \frac{2^n}{p(n)}$ for all large n, then $\mathsf{BPP}^L \subseteq \mathsf{ZPP}^L$.*

Proof. Again, the proof is very similar to the proof of the corresponding Lemma 5.18. The ZPP-machine M randomly picks a string χ_f of length m, for $m = n^c$, until it finds a string $\chi_f \in L$. As L contains at least $\frac{2^m}{p(m)}$ strings of length m, the expected run time of M until it finds a string that belongs to L is $\mathcal{O}(p(m))$. As in the proof of Lemma 5.18, if we interpret $\chi_f \in L$ as the truth-table of a Boolean function f on inputs of size $\lfloor \log m \rfloor$, we can argue that such a function f requires circuits of size at least $\Omega(m^\alpha)$ for some constant $\alpha < 1$, even if these circuits have access to L.
The function f is of sufficient hardness to construct a generator G_f^{IW97}, as stated in Theorem 5.10, to stretch $\mathcal{O}(\log m)$ random bits into m pseudorandom bits. The output of this generator is indistinguishable from random by any oracle circuit of size n with access to L. Thus, with an expected polynomial run time, we can fully derandomize the BPP-computation. □

Applying the previous lemma with an oracle A complete under linear-time reductions for $\mathsf{DSpace}[n]$ yields the following corollary.

Corollary 5.28. $\mathsf{BPP}^{R_{\mathrm{KS}}} = \mathsf{ZPP}^{R_{\mathrm{KS}}}$.

Placing PSPACE into $\mathsf{BPP}^{R_{\mathrm{Kt}}}$ requires the use of the generator G_f^{IW98} from [IW98] that is built on a uniform hardness assumption. We will also need the following Theorem of [TV02].

Theorem 5.29. *There is a problem in $\mathsf{DSpace}[n]$ that is downward self-reducible, random self-reducible, and hard for PSPACE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reductions.*

Proof of Theorem 5.26. Let $f \in \mathsf{DSpace}[n]$ be a function that is downward self-reducible, random self-reducible, and that is hard for PSPACE. The existence of such a function is guaranteed by Theorem 5.29. First, we will show that $f \in \mathsf{BPP}^{R_{\mathrm{KS}}}$.

Consider $G_f^{IW98} : \{0,1\}^{n^{\frac{1}{2}}} \mapsto \{0,1\}^n$. Theorem 5.9 states that the output of this generator can be computed in space $\mathcal{O}(n^{\frac{1}{2}})$ given oracle access to f. However, f can be computed in linear space and thus the generator can compute the function values of f directly without significantly increasing its space requirements. Therefore, all the strings in the range of G_f^{IW98} have small space-bounded Kolmogorov complexity. More precisely, $\mathrm{KS}(x) \leq \mathcal{O}(|x|^{\frac{1}{2}})$, for any $x = G_f^{IW98}(s)$, for some seed s. This implies that $G_f^{IW98}(s) \notin R_{\mathrm{KS}}$ for all s. Hence, R_{KS} distinguishes the output of G_f^{IW98} from random strings. This suffices to apply Theorem 5.9. There is a probabilistic procedure with oracle R_{KS} that on input x outputs $f(x)$ with probability at least $\frac{2}{3}$. Therefore, $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{R_{\mathrm{KS}}}$. Now, we can use Corollary 5.28 to derandomize the $\mathsf{BPP}^{R_{\mathrm{KS}}}$ computation to obtain $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{R_{\mathrm{KS}}}$. □

The above proof of Theorem 5.26 first proves $\mathsf{PSPACE} \in \mathsf{BPP}^{R_{\mathrm{KS}}}$ and then derandomizes the BPP computation with a zero-error probabilistic computation. It would be desirable to strengthen the result and obtain a *deterministic* polynomial time algorithm with oracle R_{KS} for problems in PSPACE. The main obstacle is the question of how to deterministically generate a sufficiently hard string, such that the BPP computation can be derandomized. Inspired by a construction due to [BFNV03], we can utilize that symmetry of information holds for resource-unbounded Kolmogorov complexity. This enables us to use R_{K} as an oracle to incrementally build a string with high K-complexity.

Lemma 5.30. $\mathsf{BPP}^{R_{\mathrm{K}}} = \mathsf{P}^{R_{\mathrm{K}}}$

Proof. For any $m = n^{O(1)}$, we first show how to construct a string z of length m with $\mathrm{K}(z) \geq \frac{1}{2}|z|$ using a polynomial time computation that has access to oracle R_{K}. By Theorem 3.28 there is a constant c_1, such that for any strings z and y, $\mathrm{K}(zy) \geq \mathrm{K}(z) + \mathrm{K}(y|z) - c_1 \log |zy|$.

We build z inductively and start with $z = \lambda$. Assume that $\mathrm{K}(z) \geq \frac{1}{2}|z|$. Try all strings

y of length $2c_1 \log m$, and use the oracle R_{K} to see if $\mathrm{K}(zy) \geq \frac{1}{2}|zy|$. We are guaranteed to find such a y, since $\mathrm{K}(y|z) \geq |y|$ holds for some y. For any such y, we can bound the complexity $\mathrm{K}(zy) \geq \mathrm{K}(z) + \mathrm{K}(y|z) - c_1 \log|zy| \geq \frac{1}{2}|z| + |y| - c_1 \log|zy| \geq \frac{1}{2}|zy|$. Repeat with $z = zy$ until $|z| = m$.

Note that any function $f : \{0,1\}^{\log m} \mapsto \{0,1\}$ that is computable by a circuit C of size $m^{\frac{1}{2}}$ with oracle R_{K} has Kolmogorov complexity at most $\mathcal{O}(m^{\frac{1}{2}} \log m)$. (We consider the Kolmogorov complexity of a function f to be the Kolmogorov complexity of its characteristic sequence χ_f.) This is because a circuit of size at most $m^{\frac{1}{2}}$ makes oracle queries of size at most $m^{\frac{1}{2}}$. The set R_{K} is recursively-enumerable. Thus, we only need to specify how many of strings of length at most $m^{\frac{1}{2}}$ are contained in R_{K} in order to compute the characteristic function of R_{K}. This can be done using $m^{\frac{1}{2}} + 1$ bits. Hence, f can be described by these $m^{\frac{1}{2}} + 1$ bits plus the description of the circuit C plus some constant. Thus, $\mathrm{K}(f) \leq \mathcal{O}(m^{\frac{1}{2}} \log m)$.

It follows that the function f given by the characteristic sequence z requires R_{K} oracle circuits of size at least $m^{\frac{1}{2}}$. By Theorem 5.10, we can use z to derandomize $\mathsf{BPP}^{R_{\mathrm{K}}}$ computations. □

If we combine Lemma 5.30 with the fact that the argument of the proof of Theorem 5.26 also shows $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{R_{\mathrm{K}}}$, we obtain the following theorem.

Theorem 5.31. *R_{K} is hard for PSPACE under $\leq_{\mathrm{T}}^{\mathsf{P}}$ reductions.*

It is not clear if the same technique can be utilized to show $\mathsf{PSPACE} \subseteq \mathsf{P}^{R_{\mathrm{KS}}}$. It is not known if symmetry of information holds for KS. (It does not hold for Kt, →Theorem 3.33.) It is not even clear whether the weaker (but sufficiently strong) statement holds that there are constants c, ε, such that for every n and every x, there is a string y of length $|y| = c \log n$, such that $\mathrm{KS}(xy) \geq \mathrm{KS}(x) + \varepsilon|y|$.

As a lower bound we can show that $p(n)$ bits of advice, for a fixed polynomial p, is not sufficient to reduce every set in PSPACE to R_{KS} in space $p(n)$.

Theorem 5.32. *For every k, there is a set $L \in \mathsf{PSPACE}$, such that L is not truth-table reducible to R_{KS} in space n^k with n^k bits of advice.*

Again, we rely on the fact that the length of queries to R_{KS} that the reducing machine can ask is bounded by a fixed polynomial.

Lemma 5.33. *Let $k > 0$ be fixed. Assume L is truth-table reducible to R_{KS} in space $s(n) = n^k$ by an oracle machine M_1 that uses n^k bits of advice. There is a machine M_2 with oracle R_{KS} that decides L in space $\mathcal{O}(n^{k+1})$ given n^k bits of advice and that only asks queries q of length $|q| \leq 2|x|^{k+1}$.*

Proof. The proof is identical to the proof of Lemma 5.23. □

Proof of Theorem 5.32. Let $(M_i)_{i\in\mathbb{N}}$ be an enumeration of all oracle Turing machines with an advice tape, running in space $s(n) = n^k$. We define a set $L \in \mathsf{PSPACE}$ that cannot be decided by a machine in space $\mathcal{O}(n^k)$ that queries the oracle R_{KS} non-adaptively, even when given n^k bits of advice.
Let some sufficiently large n be fixed. We will diagonalize against the first n Turing machines with oracle R_{KS} and against all possible advice strings. We define a sequence $A_0, \ldots, A_{2^n}$ of sets as follows. The initial set is given by

$$A_0 = \left\{ (M_j, a) \;\middle|\; 1 \leq j \leq n, a \in \{0,1\}^{n^k} \right\}$$

Note $|A_0| \leq n \cdot 2^{n^k}$.
Let $x_1, \ldots, x_{2^n}$ be all the strings of length n. We define L inductively as follows.

$$\begin{aligned} x_i \in L \iff & \text{for at least } \tfrac{1}{2} \text{ of the pairs } (M_j, a) \in A_{i-1}, \text{ for } 1 \leq j \leq n, \\ & M_j^{R_{\mathrm{KS}}}(x_i) = 0 \text{ with advice } a \quad . \end{aligned}$$

Note that $x_i \in L$, if $|A_{i-1}| = 0$. Now we can define the subsequent sets A_i as

$$A_i = \big\{ (M_j, a) \in A_{i-1} \;\big|\; x_i \in L \iff M_j^{R_{\mathrm{KS}}}(x_i) = 1, \text{ with advice } a, \text{ for } 1 \leq j \leq n \big\} \quad .$$

Intuitively, the set A_i contains all the machine, advice pairs that agree with L on the strings $x_1, \ldots, x_i$. Note that by definition of L, at most half of the machine, advice pairs in A_{i-1} agree with L on x_i, hence

$$|A_i| \leq \tfrac{1}{2} \cdot |A_{i-1}| \quad , \quad \text{therefore} \quad |A_i| \leq \frac{1}{2^i} \cdot |A_0| \quad .$$

Let $c = \lceil \log n^k \rceil + \lceil \log(n+1) \rceil$. Then

$$\begin{aligned} |A_{n^c}| &\leq \frac{1}{2^{n^c}} \cdot |A_0| \\ &\leq \frac{1}{(n+1) \cdot 2^{n^k}} \cdot |A_0| \\ &\leq \frac{1}{(n+1) \cdot 2^{n^k}} \cdot (n \cdot 2^{n^k}) \\ &< 1 \end{aligned}$$

Since $|A_{n^c}|$ must be integral, we know that A_{n^c} is empty, i.e. no machine, advice pair agrees with L on all strings $x_1, \ldots, x_{n^c}$. In order to obtain the conclusion that L cannot be reduced to R_{KS} in time n^k with n^k bits of advice by a truth-table reduction, we just need to note that every oracle Turing machine M_j running in space 2^{n^k} relative to oracle R_{KS}, agrees with L on at most finitely many input lengths, even when provided with advice strings of length n^k.

In order to show that $L \in \mathsf{PSPACE}$, note the following. If $i > n^c$, we know that $x_i \in L$. In order to decide $L(x_i)$ for $i \leq n^c$, we first need to recursively compute the sets $A_{i-1}, \ldots, A_0$. Note that the sets A_i are too big to be explicitly written down. Rather,

we need to recompute them as needed. Then, for each pair $(M_j^{R_{\mathrm{KS}}}, a) \in A_{i-1}$, we have to simulate $M_j^{R_{\mathrm{KS}}}(x_i, a)$. Each machine runs in space $|x|^k$ and asks queries of length at most $2|x|^{k+1}$. We know that $R_{\mathrm{KS}} \in \mathsf{DSpace}[n]$ and thus each query can be answered in space $\mathcal{O}(|x|^{k+1})$. Thus the space required to compute $M_j^{R_{\mathrm{KS}}}(x_i, a)$ can be generously bounded from above by $|x|^{k^3}$. The space required to decide membership for A_i is thus $\log(n \cdot 2^{n^k}) \cdot n^{k^3} = n^{\mathcal{O}(1)}$ plus the space to decide membership in A_{i-1}. Resolving this recurrence still keeps the space bound polynomial. Hence, $L \in \mathsf{PSPACE}$. □

5.6 Completeness results for KNt

In this subsection we will show that R_{KNt} is complete for $\mathsf{NEXP}/_{\mathsf{poly}}$. The proof uses the same technique as for the completeness result of R_{Kt} and R_{KS}. However, in order to argue that the output of a generator G_f^{BFNW} based on a function $f \in \mathsf{NEXP}/_{\mathsf{poly}}$ has low KNt complexity, we need to rely on the following fact that $\mathsf{NEXP}/_{\mathsf{poly}}$ is closed under complement. While it is not known to us if this Lemma has been formally published, it was communicated to us through [vM03, For04].

Lemma 5.34 (Folklore). $\mathsf{NE}/_{\mathsf{lin}} = \mathsf{co\text{-}NE}/_{\mathsf{lin}}$

Using a standard padding argument, this lemma implies the following Corollary.

Corollary 5.35. $\mathsf{NEXP}/_{\mathsf{poly}} = \mathsf{co\text{-}NEXP}/_{\mathsf{poly}}$

Proof of Lemma 5.34. We need to show that for every $L \in \mathsf{NE}$ there is an advice function $a : \mathbb{N} \mapsto \{0,1\}^*$ with $|a(n)| = \mathcal{O}(n)$ and deterministic machine M running in linear time, such that $x \notin L$ if and only if there is a string $w \in \{0,1\}^{\mathcal{O}(2^n)}$ for which $M(a(|x|), x, w)$ accepts.

Let us define the advice function as $a(n) = \mathrm{bin}(\mathrm{cens}_L(n))$. It indicates how many strings of length n the set L contains. Since $L \in \mathsf{NE}$, there is a deterministic machine M' running in linear time, such that $x \in L$ if and only if there is a witness $w \in$

$\{0,1\}^{\mathcal{O}(2^n)}$ such that $M'(x,w)$ accepts. Now define the machine M as follows. The machine only accepts inputs of the form $\langle a, x, w' \rangle$ for $w' = \langle y_1, w_1, y_2, w_2, \ldots, y_a, w_a \rangle$. Given such an input M checks if $M'(y_i, w_i) = 1$ for each $1 \leq i \leq a$. If this is the case, the machine M accepts if and only if $x \neq y_i$ for each i and $y_i \neq y_j$ for $i \neq j$. It is straightforward to verify that there is a $w' \in \{0,1\}^{2^{\mathcal{O}(n)}}$ such that $M(a, x, w')$ accepts if and only $x \notin L$. The run time is bounded by $2^{\mathcal{O}(|x|)}$. □

As it is not clear whether NEXP is closed under complement, we are only able to obtain a completeness result for $\mathsf{NEXP}/\mathsf{poly}$ but not for NEXP.

Theorem 5.36. *R_{KNt} is complete for $\mathsf{NEXP}/\mathsf{poly}$ under $\leq_{\mathrm{tt}}^{\mathsf{P}/\mathsf{poly}}$ reductions.*

In order to use Theorem 5.12 to show that R_{KNt} is complete for $\mathsf{NEXP}/\mathsf{poly}$, we have to argue that there is as a function A that is complete for NE and that is also PSPACE-robust.

Lemma 5.37. *There is a set A that is complete for NE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ and that is PSPACE-robust.*

Proof. Let A be complete for NE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$. We will argue that $\mathsf{P}^A = \mathsf{PSPACE}^A$.
Let $B \in \mathsf{PSPACE}^A$ be decidable by the machine M with oracle A in polynomial space. Assume that on all inputs of length n the machine M only makes queries of length at most n^k. Let $a(n) = |A^{\leq n^k}|$ indicate the number of strings of length at most n^k contained in A.
First we will argue that $a(n)$ can be computed in polynomial time, given access to A. To see this, consider the set $A' = \{\langle m, a \rangle \mid |A^{\leq m}| \geq a\}$. It is easy to argue that $A' \in \mathsf{NEXP}$, since the nondeterministic machine can just guess a strings $y_1, \ldots, y_a$ of length at most m and a possible witnesses $w_1, \ldots, w_a$ and then verify if for each i the string w_i is a witness for y_i. The exact value of $a(n)$ can now be computed with n^k queries to A' using binary search. Since A is complete for NE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reductions, it

is complete for NEXP under $\leq_{\mathrm{m}}^{\mathsf{poly}}$ reductions. Thus, all queries to A' can be reduced to queries to A in polynomial time.

Next, consider the following nondeterministic machine N. On input $\langle x, a\rangle$ the machine N guesses a set $S = \{y_1, \ldots, y_a\}$ of a strings of length n^k and a possible witnesses $w_1, \ldots, w_a$, and it verifies if for each i the string w_i certifies that $y_i \in A$. If this succeeds the machine N simulates the computation of M^S, and N accepts if M^S accepts. Note that M only asks queries of length n^k. Thus if $a = a(n)$ and consequently $S = A^{=n^k}$, the machine N accepts, if and only if $M^A(x)$ accepts, if and only if $x \in B$. Further note that $L(N) \in \mathsf{NEXP}$. And since A is complete for NE under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ reductions, we can conclude that $L(N) \in \mathsf{P}^A$.

We argued that $a(n)$ can be computed in polynomial time given access to A. Furthermore we argued that $\langle x, a(n)\rangle \in L(N)$ if and only if $x \in B$ and also that $L(N) \in \mathsf{P}^A$. Therefore $B \in \mathsf{P}^A$. □

Now we can prove that R_{KNt} is complete for $\mathsf{NEXP}/_{\mathsf{poly}}$.

Proof of Theorem 5.36. The proof is essentially identical to the proof of Theorem 5.16. Lemma 5.37 guarantees the existence of a set A that is complete for NE. under $\leq_{\mathrm{m}}^{\mathsf{lin}}$ and that is PSPACE-robust. Note that $\mathrm{cens}_{R_{\mathrm{KNt}}}(n) \geq 2^n - 2^{\frac{n}{2}}$. Thus R_{KNt} has more than polynomial density. Next, recall (→Theorem 3.22) that there is a constant $c > 0$, such that $c \cdot \mathrm{KNt}(x) > \mathrm{KT}^A(x)$. Therefore, for every $x \in R_{\mathrm{KNt}}$, it holds that $\mathrm{KT}^A(x) \geq (\frac{1}{2c}|x|)$. Thus any $\varepsilon < 1$ yields $\mathrm{KT}^A(x) \geq |x|^\varepsilon$ for all sufficiently long x. Now Theorem 5.12 directly gives the desired result. □

In contrast with R_{Kt} we are able to provide an unconditional lower bound on the complexity of R_{KNt}.

Theorem 5.38. $R_{\mathrm{KNt}} \notin \mathsf{ZPP}$.

Proof. If $R_{\mathrm{KNt}} \in \mathsf{ZPP}$, then Theorem 5.36 yields $\mathsf{NEXP}/_{\mathsf{poly}} \subseteq \mathsf{P}^{\mathsf{ZPP}}/_{\mathsf{poly}} = \mathsf{P}/_{\mathsf{poly}}$. Thus $\mathsf{NEXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$ and by the results of [IKW02] $\mathsf{NEXP} \subseteq \mathsf{P}/_{\mathsf{poly}}$ yields

$\mathsf{NEXP} \subseteq \mathsf{MA} \subseteq \mathsf{PSPACE}$. On the other hand, the proof of Theorem 5.26 also yields $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{R_{\mathrm{KNt}}}$. The assumption $R_{\mathrm{KNt}} \in \mathsf{ZPP}$ simplifies this to $\mathsf{PSPACE} \subseteq \mathsf{ZPP}$ and therefore $\mathsf{NEXP} \subseteq \mathsf{ZPP}$ However, this contradicts the nondeterministic time-hierarchy theorem which states $\mathsf{NEXP} \nsubseteq \mathsf{NP}$. □

5.7 Complexity of R_{KT}

The previous subsections paint an illuminating picture about the hardness of sets with high Kt, KS, and KNt complexity for PSPACE, EXP, and larger complexity classes. In this section, we explore what these techniques have to say about the hardness of R_{KT}, a set in coNP. We are not able to show completeness of R_{KT} for coNP, but we can show the hardness of R_{KT} under randomized polynomial-time reductions for problems that are thought to be NP-intermediate: Discrete Log, Integer Factorization, and certain lattice problems.

Note that the set R_{KT} seems closely related to the set MCSP defined in [KC00] as

$$\mathrm{MCSP} = \big\{ \langle \chi_f, s \rangle \;\big|\; \chi_f \text{ is the truth-table of a function } f \text{ and } s \text{ is the size of smallest circuit computing } f \big\} \quad .$$

Although we do not know of efficient reductions between R_{KT} and MCSP in either direction, all our hardness results for R_{KT} also hold for MCSP. Based on a connection with natural proofs, [KC00] showed that if MCSP is in P then, for any $\varepsilon > 0$, there is a randomized algorithm running in time 2^{n^ε} that factors Blum integers well on the average. Our results imply that if MCSP is in P then there is a randomized polynomial-time algorithm that factors arbitrary integers.

The known hardness versus randomness tradeoffs for G_f^{HILL} differ in two relevant respects from those provided by the Nisan-Wigderson style generators.

First, breaking G_f^{HILL} only lets us invert f on a non-negligible fraction of the inputs,

but not necessarily on all inputs. The implicit or explicit random self-reducibility of the problems considered in the previous completeness results allowed us there to make the transition from computing f on a non-negligible fractions of inputs to computing it on all inputs. However, unlike for EXP and PSPACE, there are no NP-complete problems that are known to be random self-reducible. In fact, there provably are no non-adaptively random self-reducible NP-complete sets unless the polynomial-time hierarchy collapses [FF93].

Nevertheless, for some specific NP-intermediate problems h and polynomial-time computable functions f (where h may or may not coincide with f^{-1}), a worst-case to average-case connection *is* known. That is, inverting f on a non-negligible fraction of the inputs allows one to compute h efficiently on any input. We are able to prove hardness of R_{KT} for such problems.

The second difference with the previous hardness results is the fact that the uniform hardness versus randomness tradeoffs in [HILL99] are as strong as the nonuniform ones. Therefore, we only need to consider uniform reductions here.

Theorem 5.11 states that if there exists a distinguisher with access to an oracle L that distinguishes the output of G_f^{HILL} from the uniform distribution, then oracle access to L suffices to invert f on a polynomial fraction of the inputs. We now argue that such a distinguisher exists in the case where L denotes R_{KT} or any set of polynomial density that contains no strings of small KT complexity.

Theorem 5.39. *Let L be a language of polynomial density such that, for some $\varepsilon > 0$, for every $x \in L$, $\mathrm{KT}(x) \geq |x|^{\varepsilon}$. Let $f(y,x)$ be computable uniformly in time polynomial in $|x|$. There exists a polynomial-time probabilistic oracle Turing machine N and polynomial q such that for any n and y:*

$$\Pr_{x \in U_n, s}\left[\, f(y, N^L(y, f(y,x), s)) = f(y,x) \,\right] \geq \frac{1}{q(n)} \quad ,$$

where s denotes the internal coin flips of N.

Proof. Let $G_y(x)$ denote $G_{f_y}^{HILL}(x)$. As in [RR97], we use the construction of [GGM86] to modify G_y and obtain a pseudorandom generator G'_y producing longer output. Specifically, the proof of Theorem 4.1 of [RR97] constructs $G'_y : \{0,1\}^n \mapsto \{0,1\}^{2^k}$, and shows that if $z = G'_y(x)$, then z can be computed by circuits of size $(n+k)^{O(1)}$. We can pick $k = O(\log n)$ such that for each x and polynomially bounded y, $\mathrm{KT}(G'_y(x)) < |G'_y(x)|^{\varepsilon}$. Thus L is a statistical test that accepts many random strings of length $|x|^{O(1)}$, but rejects all pseudorandom strings. As in [RR97], this gives us a probabilistic oracle machine M using L that distinguishes G_y from the uniform distribution. We then apply Theorem 5.11. □

We now apply Theorem 5.39 to obtain our hardness results for R_{KT}. As will be clear from the proofs, R_{KT} can be replaced by any suitably dense language containing no strings of KT-complexity less than n^{ε} for some $\varepsilon > 0$, e.g., MCSP.

We first consider the Discrete Logarithm Problem, which takes as input a triple (p, g, z) where p is a prime number, $0 < g < p$, and $0 < z < p$, and outputs a positive integer i such that $g^i \equiv z \bmod p$, or 0 if there is no such i. We have:

Theorem 5.40. *The Discrete Logarithm Problem is computable in* $\mathsf{BPP}^{R_{\mathrm{KT}}}$.

Proof. On input (p, g, z), we first check if p is prime. This takes polynomial time [AKS02]. Let n be the number of bits in p, and let y denote the pair (p, g). Consider the function $f(y, x) = g^x \bmod p$. Theorem 5.39 with $L = R_{\mathrm{KT}}$ gives us a polynomial-time oracle Turing machine N such that for some polynomial q and all p and g, for randomly chosen x and s, the machine $N^{R_{\mathrm{KT}}}(p, g, g^x, s)$ produces an output i such that $g^i \equiv g^x \bmod p$ with probability at least $\frac{1}{q(n)}$.

Now we make use of the self-reducibility properties of the Discrete Log. In particular, on our input (p, g, z), we choose many more than $q(n)$ values v at random and run algorithm N on input $(p, g, zg^v \bmod p)$. If z is in the orbit of g, then with high

probability at least one of these trials will return a value u such that $g^u = zg^v \bmod p$, which means that we can pick $i = u - v$ and obtain $z \equiv g^i \bmod p$.
On the other hand, if none of the trials is successful, then with high probability z is not in the orbit of g and the algorithm should return 0. □

We are not able to improve our reduction from a BPP-reduction to a ZPP-reduction. We note that, for inputs (p, g, x) such that x is in the orbit of g, which is the usual class of inputs for which the discrete log is of interest, we do have ZPP-like behavior, since we can check whether the number i we obtain satisfies $g^i \equiv x \bmod p$. However, when x is not in the orbit of g, we obtain no *proof* of this fact – merely strong evidence. The next problem we consider is an example of a problem, where the function h we wish to compute is not directly the inverse of the function f on which the generator is based.

Theorem 5.41. *Integer Factorization is computable in* $\mathsf{ZPP}^{R_{\mathrm{KT}}}$.

Proof. Consider Rabin's candidate one-way function $f(y, x) = x^2 \bmod y$, where $0 \leq x < y$. Theorem 5.39 with $L = R_{\mathrm{KT}}$ gives us a probabilistic polynomial-time procedure with oracle access to R_{KT} that, for any fixed y, finds an inverse of $f(y, x)$ for a fraction at least $\frac{1}{|y|^{O(1)}}$ of the x's with $0 \leq x < y$. Rabin [Rab79] showed how to use such a procedure and some randomness to efficiently find a nontrivial factor of y in case y is composite. This leads to a $\mathsf{BPP}^{R_{\mathrm{KT}}}$ algorithm for factoring. Since primality is in P [AKS02], we can avoid errors and obtain the promised $\mathsf{ZPP}^{R_{\mathrm{KT}}}$ factoring algorithm. □

Since Ajtai's seminal paper [Ajt96], several worst-case to average-case connections have been established for lattice problems. We will exploit these next.
We first review some lattice terminology. We refer to [Cai99b] for more background. Given a set B of n linearly independent vectors $b_1, \ldots, b_n$ over $\mathbb{R}^n$, the set $L(B) = \{ \sum_{i=1}^{n} z_i b_i \,|\, z_i \in \mathbb{Z} \}$ is called the lattice spanned by the basis B. We consider the

usual notion of length of a vector v, $|v| = \sqrt{\sum v_i^2}$, and define the length of a set of vectors as the length of a longest vector in that set. For a lattice $L(B)$, $\lambda_i(B)$, $1 \leq i \leq n$, denotes the length of a shortest set of i linearly independent vectors from $L(B)$. The covering radius of $L(B)$, $\rho(B)$, is defined as the smallest ρ such that the union of all spheres of radius ρ centered at the points of $L(B)$ covers the entire space $\mathbb{R}^n$.

Applying our technique to Ajtai's worst-case to average-case connections and their subsequent improvements and extensions [CN97, Mic02], we obtain the following hardness results for R_{KT}.

Theorem 5.42. *For every $\varepsilon > 0$, we can solve each of the following problems in $\mathsf{BPP}^{R_{\mathrm{KT}}}$: Given a basis $B \subseteq \mathbb{Z}^n$ (and a vector $t \in \mathbb{Z}^n$), find*

- *(Shortest Independent Vector Problem) a set of n linearly independent vectors in $L(B)$ of length at most $n^{3+\varepsilon} \cdot \lambda_n(B)$,*
- *(Shortest Basis Problem) a basis for $L(B)$ of length at most $n^{3.5+\varepsilon} \cdot \lambda_n(B)$,*
- *(Length of Shortest Vector Problem) a value $\tilde{\lambda}$, such that $\lambda_1(B) \leq \tilde{\lambda} \leq \omega(n^{3.5} \log n) \cdot \lambda_1(B)$,*
- *(Unique Shortest Vector Problem) a shortest vector in $L(B)$ in case $\lambda_2(B) > n^{4+\varepsilon} \cdot \lambda_1(B)$,*
- *(Closest Vector Problem) a vector $u \in L(B)$ such that $|u - t| \leq n^{3.5+\varepsilon} \cdot \lambda_n(B)$, and*
- *(Covering Radius Problem) a value $\tilde{\rho}$, such that $\rho(B) \leq \tilde{\rho} \leq \omega(n^{2.5} \log n) \cdot \rho(B)$.*

In order to construct the reduction for SIVP, we will use the following worst-case to average-case connection.

Theorem 5.43 ([Ajt96, CN97]). *For any $\varepsilon > 0$, there exist functions $q(n) = n^{O(1)}$ and $m(n) = n \log^2 q$ with q a power of 2 such that the following holds. For every $c > 0$, if there is probabilistic algorithm A, such that, with probability at least $1/n^c$ (over $M \in \mathbb{Z}_q^{n \times m}$ and over the random choices of A), $A(M)$ outputs a nonzero vector*

$u \in \mathbb{Z}^m$ such that $|u| \leq m$ and $Mu \equiv 0 \bmod q$, then there exists a BPP^A-algorithm A' that given any basis $B \subseteq \mathbb{Z}^n$, outputs with high probability a set of n linearly independent vectors in $L(B)$ of length at most $n^{3+\varepsilon} \cdot \lambda_n(B)$.

Now we can prove Theorem 5.42

Proof of 5.42. Let $q(n)$ and $m(n)$ be as in Theorem 5.43. Consider the collection of functions $f_{q,m} : \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \mapsto \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$ defined by $f_{q,m}(M, v) = (M, Mv \bmod q)$. Letting $y = (q, m)$ and $x = (M, v)$, the function $f_y(x)$ is computable uniformly in time $|x|^{O(1)}$. By Theorem 5.39, there exists a polynomial-time probabilistic algorithm N with oracle access to R_{KT} that computes a preimage of $f_{q,m}(M, v)$ for at least a $\frac{1}{n^{O(1)}}$ fraction of the inputs (M, v).

For any fixed M, f maps to at most q^n different values. Thus there can be at most q^n vectors v, such that $f(M, v) \neq f(M, v')$ for all $v' \neq v$. In other words, for a fraction at least $1 - \frac{q^n}{2^m} > 1 - O(\frac{1}{2^n})$ of the vectors v, there is a $v' \neq v$ with $f(M, v') = f(M, v)$. Therefore, if we pick M and v uniformly at random, and run N to invert $f(M, Mv \bmod q)$, with probability at least $1/n^{O(1)}$ we obtain a vector $v' \in \mathbb{Z}^m$ such that $v' \neq v$ and $Mv \equiv Mv' \bmod q$.

Setting $u = v - v'$ yields a nonzero vector in $\mathbb{Z}^m$ satisfying $|u| \leq m$ and $Mu \equiv 0 \bmod q$. This vector u is the output of the algorithm A on input M. Applying Theorem 5.43 to A yields the $\mathsf{BPP}^{R_{\mathrm{KT}}}$ algorithm for SIVP.

The reductions for SBP, Unique-SVP, and CVP follow from the one for SIVP by arguments given in [CN97, Cai01]. The results for LSVP and CRP are obtained in a similar way based on the variants of the candidate one-way function f and the worst-case to average-case connection corresponding to Theorem 5.43 presented in [Mic02]. □

Given our inability to prove that R_{KT} is coNP-complete, one may wonder whether R_{KT} is in NP. If it is, then this would provide a dense combinatorial property in NP

that is useful against $\mathsf{P}/_{\mathsf{poly}}$, contrary to a conjecture of Rudich [Rud97]. We can also show the following.

Theorem 5.44. *If* R_{KT} *is in* NP*, then* $\mathsf{MA} = \mathsf{NP}$*.*

Proof. It is shown in [IKW02] that if an NP machine can, on input of length n, find the truth table of a function of size $n^{O(1)}$ with large circuit complexity, then $\mathsf{MA} = \mathsf{NP}$. Certainly this is easy if R_{KT} is in NP. □

This observation (similar to ones in [IKW02]) can not be taken as evidence that $R_{\mathrm{KT}} \notin \mathsf{NP}$, since many conjecture that MA is equal to NP. However, it does show that proving R_{KT} in NP would require non-relativizing proof techniques.

References

[ABK+02] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 669–678, November 16–19 2002.

[ABK03] Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the K-random strings? In *Proceedings of the of Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *Lecture Notes in Computer Science*, pages 584–595, 2003.

[ACGS84] W. Alexi, B. Chor, O. Goldreich, and Schnorr Schnorr. RSA/rabin bits are $1/2 + 1/poly(\log N)$ secure. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 449–457, October 1984.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 99–108, 1996.

[AKRR03] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. Derandomization and distinguishing complexity. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 209–220, July 2003.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Available at: http://www.cse.iitk.ac.in/primality.pdf, 2002.

[All89] Eric Allender. Some consequences of the existence of pseudorandom generators. *Journal of Computer and System Sciences*, 39:101–124, 1989.

[All92] Eric Allender. Applications of time-bounded kolmogorov complexity in complexity theory. In *Osamu Watanabe (Ed.), Kolmogorov Complexity and Computational Complexity*, pages 4–22. Springer, 1992.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Proceedings of the Conference on Foundation of Software Technology and Theo. Comp. Sci. (FST&TCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 1–15, 2001.

[BDG95] José Balcázar, Josep Diaz, and Joaquim Gabarro. *Structural Complexity I.* Springer, Berlin, 1995.

[BF95] Harry Buhrman and Lance Fortnow. Distinguishing complexity and symmetry of information. Technical Report TR-95-11, Department of Computer Science, University of Chicago, November 08 1995. Thu, 30 Nov 1995 22:05:46 GMT.

[BFL91] Lazlo Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[BFL02] H. Buhrman, L. Fortnow, and S. Laplante. Resource-bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2002.

[BFNV03] Harry Buhrman, Lance Fortnow, Ilan Newman, and Nikolai Vereshchagin. Increasing complexity. Unpublished manuscript, 2003.

[BFNW93] Lazlo Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BG70] R. Book and S. Greibach. Quasi-realtime languages. *Mathematical Systems Theory*, 4:97–111, 1970.

[BGS75] Baker, Gill, and Solovay. Relativizations of the $\mathsf{P} =?\ \mathsf{NP}$ question. *SICOMP: SIAM Journal on Computing*, 4, 1975.

[BIS88] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. In *Proceedings of the IEEE Conference on Structure in Complexity Theory*, pages 47–59. IEEE Computer Society Press, 14–17 June 1988.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BM95] Harry Buhrman and Elvira Mayordomo. An excursion to the Kolmogorov random strings. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory (SCTC '95)*, pages 197–205, 1995.

[BT01] Harry Buhrman and Leen Torenvliet. Randomness is hard. *SIAM Journal on Computing*, 30(5):1485–1501, 2001.

[Cai99a] Jin-Yi Cai. Applications of a new transference theorem to Ajtai's connection factor. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 205–214, 1999.

[Cai99b] Jin-Yi Cai. Some recent progress on the complexity of lattice problems. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 158–179, 1999.

[Cai01] Jin-Yi Cai. On the average-case hardness of CVP. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 308–319, 2001.

[Cha66] Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):547–569, October 1966.

[Cha69] Gregory J. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16:145–159, 1969.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[CN97] Jin-Yi Cai and A. P. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 468–477, 1997.

[DK00] Ding-Zhou Du and Ker-I Ko. *Theory of Computational Complexity*. Wiley-Interscience, 2000.

[FF93] Joan Feigenbaum and Lance Fortnow. On the random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993.

[For01] Lance Fortnow. Kolmogorov complexity. In R. Downey and D. Hirschfeldt, editors, *Aspects of Complexity, Minicourses in Algorithmics, Complexity, and Computational Algebra*, volume 4 of *Logic and Its Applications*. de Gruyter, 2001.

[For04] Lance Fortnow. Kolmogorov complexity. Entry for Friday, January 30, 2004, in Fortnow's Computational Complexity Weblog, 2004. Available at http://fortnow.com/lance/complog/archive/2004_01_25_archive.html.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[GKL90] O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom generators. In *Advances in Cryptology (CRYPTO '88)*, pages 146–162, Berlin - Heidelberg - New York, August 1990. Springer.

[Har83] Juris Hartmanis. Generalized kolmogorov complexity and the structure of feasible computations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 439–445, 1983.

[HILL99] Johan Håstad, Russel Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285–306, 1965.

[HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[IKW02] Russel Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65, 2002.

[ISW99] Russel Impagliazzo, Ronen Shaltiel, and Avi Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 181–190, 1999.

[IT89] Russel Impagliazzo and Gábor Tardos. Decision versus search problems in super-polynomial time. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1989.

[IW97] Russel Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the ACM Symposium on Theory of Computing (STOC) '97*, pages 220–229, 1997.

[IW98] Russel Impagliazzo and Avi Wigderson. Randomness vs. time: derandomization under a uniform assumption. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 734–743, 1998.

[KC00] V. Kabanets and J.-Y. Cai. Circuit minimization problem. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

[Ko86] Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theoretical Computer Science*, 48(1):9–33, 1986.

[Kol68] Andrey N. Kolmogorov. Logical basis for information theory and probability theory. *IEEE Transactions on Information Theory*, 14(5):662–664, 1968.

[Kum96] Martin Kummer. On the complexity of random strings. In *Proceedings of the of Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1046 of *Lecture Notes in Computer Science*, pages 25–36, 1996.

[KvM02] Klivans and van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31, 2002.

[Lev84] Leonid Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 363–365, Providence, Rhode Island, 6–8 May 1985.

[LM93] Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. *Information Processing Letters*, 46(2):95–100, May 1993.

[Lon86] Luc Longrprè. *Resource Bounded Kolmogorov Complexity, a Link between Computational Complexity and Information Theory.* PhD thesis, Cornell University, 1986.

[LR04] Troy Lee and Andrei Romashchenko. On polynomially time bounded symmetry of information. Technical Report TR04-031, Electronic Colloquium on Computational Complexity, 2004.

[Lut87] Jack H Lutz. *Resource-Bounded Category and Measure in Exponential Complexity Classes.* Ph.D. thesis, California Institute of Technology, 1987.

[Lut97] Lutz. The quantitative structure of exponential time. In Lane A. Hemaspaandra and Alan L. Selman, editors, *Complexity Theory Retrospective II.* Springer, 1997.

[LV93] Ming Li and Paul Vitanyi. *Introduction to Kolmogorov Complexity and its Applications.* Springer, 1993.

[LV97] Ming Li and Paul Vitanyi. *An introduction to Kolmogorov complexity and its applications.* Springer, 2nd edition, 1997.

[LW95] Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. *Information and Computation*, 121(1):14–22, 15 August 1995.

[Mar66a] D.A. Martin. Completeness, the recursion theorem and effectively simple sets. *Proceding of the American Mathematical Society*, 17:838–842, 1966.

[Mar66b] P. Martin-Löf. The defintion of random sequences. *Information and Control*, 9:602–619, 1966.

[Mic02] Daniele Micciancio. Improved cryptographic hash functions with worst-case / average-case connection. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 609–618, 2002.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Pap94] Christos H. Papadimitriou. *Computatational Complexity.* Addison-Wesley, Reading, Mass., 1994.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, April 1979.

[Rab79] M. Rabin. Digitalized signatures and public-key functions as intractible as factorization. MIT Tech. Report TR-212, 1979.

[RR97] Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55:24–35, 1997.

[Rud97] S. Rudich. Super-bits, demi-bits, and $\mathrm{N}\tilde{\mathsf{P}}$/qpoly-natural proofs. In *RANDOM*, volume 1269 of *Lecture Notes in Computer Science*, 1997.

[Sha48] C. E. Shannon. The mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.

[Sol64] Ray Solomonoff. A formal theory of inductive inference, Part 1 and Part 2. *Information and Control*, 7:1–22, 224,254, 1964.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62, 2001.

[SU01] Ronen Shaltiel and Chris Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 648–657, 2001.

[TV02] Trevisan and Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the IEEE Conference on Computational Complexity*, volume 17, 2002.

[vM39] R. von Mises. *Probability, Statistics and Truth*. MacMillan, 1939. Reprint: Dover, 1981.

[vM03] Dieter van Melkebeek, 2003. Personal Communication.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity*. Springer Verlag, 1999.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

[ZL70] Alexander Zvonkin and Lenoid Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25:83–124, 1970.

Curriculum Vita

Education and Academic Employment

Jul. 1996:	"Vordiplom" (= Bachelor of Science) in Computer Science, Technical University of Berlin, Germany.
Nov. 1998:	"Diplom" (= Master of Science) in Computer Science, Technical University of Berlin, Germany.
1998-2001:	Teaching Assistant, Rutgers University, New Brunswick, NJ.
1999-2001:	Adjunct Lecturer, Rutgers University, New Brunswick, NJ.
2001-2003:	Graduate Assistant, Rutgers University, New Brunswick, NJ.
Spring 2004:	Teaching Assistant, Rutgers University, New Brunswick, NJ.
Oct. 2004:	Ph.D. in Computer Science, Rutgers University, New Brunswick, NJ.

Publications

Journal Publications

- E. Allender, M. Koucký, D. Ronneburger, S. Roy, and V. Vinay,
  Space-Time tradeoffs in the Counting Hierarchy.
  Accepted for publication in *Theory of Computing Systems.*

Conference Publications

- E. Allender, M. Koucký, D. Ronneburger, and S. Roy,
  Derandomization and Distinguishing Complexity,
  Proc. of the 18th IEEE Conf. on Computational Complexity, 2003, pp. 209-220.
- E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger,
  Power from Random Strings,
  Proc. of the 43rd Symp. on Foundations of Comp. Sci., 2002, pp. 669-678.
- E. Allender, M. Koucký, D. Ronneburger, S. Roy, and V. Vinay,
  Space-Time tradeoffs in the Counting Hierarchy,
  Proc. of the 16th IEEE Conf. on Computational Complexity, 2001, pp. 295-302.