The Non-Recursive Power of Erroneous Computation

Christian Schindelhauer[1,3] and **Andreas Jakoby**[2,3]

[1] ICSI Berkeley, 1947 Center Street, Berkeley, U.S.A. schindel@icsi.berkeley.edu[†]
[2] Depart. of Computer Science, Univ. of Toronto, Canada jakoby@cs.toronto.edu
[3] Med. Univ. zu Lübeck, Inst. für Theoretische Informatik, Lübeck, Germany

Abstract. We present two new complexity classes which are based on a complexity class $\mathcal{C}$ and an error probability function F. The first, F-Err $\mathcal{C}$, reflects the (weak) feasibility of problems that can be computed within the error bound F. As a more adequate measure to investigate lower bounds we introduce F-$\mathrm{Err}_{\mathrm{io}}\,\mathcal{C}$ where the error is infinitely often bounded by the function F. These definitions generalize existing models of feasible erroneous computations and cryptographic intractability.
We identify meaningful bounds for the error function and derive new diagonalizing techniques. These techniques are applied to known time hierarchies to investigate the influence of error bound. It turns out that in the limit a machine with slower running time cannot predict the diagonal language within a significantly smaller error prob. than $\frac{1}{2}$.
Further, we investigate two classical non-recursive problems: the halting problem and the Kolmogorov complexity problem. We present strict lower bounds proving that any heuristic algorithm claiming to solve one of these problems makes unrecoverable errors with constant probability. Up to now it was only known that infinitely many errors will occur.

1 Introduction

The answer of the question whether $\mathcal{NP}$ equals $\mathcal{P}$ is a main goal of computational complexity theory. If they differ, which is the widely proposed case, a polynomial time bounded deterministic algorithm cannot correctly decide an $\mathcal{NP}$-complete problem. Hence, the correctness of such an algorithm is the most reasonable requirement. Nevertheless, in the desperate situation where one wants to solve an infeasible problem one may accept errors, provided their influence can be controlled somehow. The quality of such an error can be exploited in more detail.

Probabilistic error: It is common sense that $\mathcal{BPP}$ can be seen as a class of efficiently solvable problems. Unlike an $\mathcal{P}$-algorithm a $\mathcal{BPP}$-algorithm can make errors, but on every input this error probability has to be bounded by $\frac{1}{3}$. So, $\mathcal{BPP}$-algorithms can be modified to decrease this error probability to an arbitrarily small non-zero polynomial. Furthermore, $\mathcal{BPP}$-algorithms solve problems for which no $\mathcal{P}$-algorithm is known yet, i.e. test of primalty.

[†] parts of this work are supported by a stipend of the "Gemeinsames Hochschulsonderprogramm III von Bund und Länder" through the DAAD.

Expected time: A valid enhancement of the notion of efficiency is to measure the expectation over the running times of different inputs according to a probability distribution over the input space. E.g. it turns out that the Davis-Putnam-algorithm [DaPu 60] solves SAT for randomly chosen Boolean formulas in polynomial expected time, if the probability distribution fulfills certain requirements [PuBr 85,SLM 92]. Note that, since this algorithm is deterministic, the probability refers only to the way of choosing the input. Probabilistic algorithms (using random bits) have been investigate in this relationship, too [Hoos 98]. An algorithm which is efficient with respect to its expected time behaviors has to compute a function always correctly, but its computation time can exceed the expected time bound for some inputs, enormously.

Average complexity classes: It turns out that complexity classes defined by expected time bounds are not closed under polynomial time bounded simulations. For this reason Levin defined *polynomial on the average* [Levi 86], a superset of expected polynomial time that initiated the average-case branch of computational complexity. Using Levin's measure for the average complexity there exists a reasonable notion of *$\mathcal{NP}$-average-case-completeness.*

Traditionally, one investigates the worst case of resources over an input length needed to solve a problem. In average case complexity theory these resources are weighted by a probability distribution over the input space. As a consequence, in worst case theory it is only necessary to consider functions $f : \mathbb{N} \rightarrow \mathbb{N}$ for an entire description of the considered complexity bound. In average case complexity theory there are a variety of ways to average over the resource function. Here it is necessary to examine pairs of resource functions $f : \Sigma^* \rightarrow \mathbb{N}$ (e.g. time) and probability distributions over the set of inputs. A variety of different concepts are investigated to define average complexity measures and corresponding classes [Levi 86,Gure 91,BCG 92,ScWa 94,CaSe 99,ReSc 96]. All these concepts have in common that the running times of all possible input values with positive weights account for the average behavior. An important result in average complexity is that if *average-$\mathcal{P}$* (Av$\mathcal{P}$) covers $\mathcal{NP}$, then $\mathcal{NE} = \mathcal{E}$ [BCG 92]. Furthermore, the fraction of non-polynomial computations of an algorithms solving an $\mathcal{NP}$-problem can be bounded under this premise.

Benign faults: Here the algorithm outputs a special symbol "?" on inputs where it fails, yet produces correct outputs in polynomial time for all other inputs. An algorithm for an $\mathcal{NP}$-problem producing only a small quantity of so-called *benign faults* can be transformed into an Av$\mathcal{P}$-algorithm [Imp2 95]. This observation is intuitively clear. Since, if an algorithm "knows" that it cannot solve a certain instance of an $\mathcal{NP}$-problem, it can use the trivial exponentially time-bounded simulation of the nondeterministic Turing-machine. If the probability for these instances is exponentially small, the resulting average-case time stays polynomial.

Similar questions were considered in [Schi 96], e.g. Schindelhauer introduces the class MedDistTime(T,F) which is strongly related to the statistical p-th quantile. Here, a machine M may violate the time bound $T(|x|)$ for at most $F(\ell)$ of the ℓ most likely inputs x. But the machine has to decide on the language correctly. This setting of correctness is equivalent to benign fault when we consider time-bounded complexity classes.

Real faults: In [ImWi 98,Schi 96,ScJa 97,Yam1 96,Yam2 96] a different approach is introduced. They investigate the number of inputs for a given machine causing an erroneous computation or breaking the time limit respectively. Note that for a fix time bound exceeding the time limit causes an error.

Yamakami proposed the notion of Nearly-$\mathcal{P}$ and Nearly-$\mathcal{BPP}$ (see [Yam1 96] and [Yam2 96]). Here the error probability has to be smaller than any polynomial in the input length. More precisely, even a polynomial number of instances (chosen according to the probability distribution) induces a super-polynomial small error bound again. Thus, Nearly-$\mathcal{P}$ and Nearly-$\mathcal{BPP}$ define reasonable efficient complexity classes if the corresponding algorithm is only used for a polynomial number of inputs for each length.

Independently, Schindelhauer et al. in [Schi 96,ScJa 97] introduced a similar approach. Based on a T-time decidable languages L and an error probability function F they investigate pairs of languages L' and probability distribution μ where for all $\ell \in \mathbb{N}$ the number of the ℓ most likely inputs $x \in \Sigma^*$ (according to μ) with $L'(x) \neq L(x)$ is bounded by $F(\ell)$.

Impagliazzo and Wigderson in [ImWi 98] investigate the relationship between $\mathcal{BPP}$ and an error complexity class called $\mathrm{HeurTime}_{\epsilon(n)}(T(n))$, where for each input length the number of unrecoverable errors of a T-time bounded algorithm is bounded by ϵ.Their main motivation for this definition is the better understanding of the relationship of $\mathcal{BPP}$, $\mathcal{P}\backslash$poly and $\mathcal{E}$.

Erroneous Computation has a practical impact in designing more efficient algorithms, see for example [Fagi 92,GeHo 94,Reif 83]. In these papers parallel algorithms, resp. circuits, for adding two large binary numbers are presented which are efficient on the average. The basic part of these strategies is a fast but erroneous algorithm with a polynomial share of inputs causing wrong outputs. This results in a double logarithmic time bound. An additional error-detection-strategy restores correctness.

Based on this work done so far, it is reasonable to extend these definitions to arbitrary classes $\mathcal{C}$ and to consider the general properties of error complexity classes.

Definition 1 *For a class $\mathcal{C}$ and a bound $F : \mathbb{N} \to [0,1]$ define the distributional complexity classes of* ***F-error bounded $\mathcal{C}$*** *and* ***infinitely often F-error bounded $\mathcal{C}$*** *as sets of pairs of languages $L \subseteq \Sigma^*$ and probability distributions $\mu : \Sigma^* \to [0,1]$ as follows:*

$$\boldsymbol{F}\text{-}\mathbf{Err}\boldsymbol{\mathcal{C}} := \{(L,\mu) \mid \exists S \in \mathcal{C}\ \forall n\ :\ \mathrm{Prob}_\mu[x \in (L \triangle S) \mid x \in \Sigma^n] \ \leq\ F(n)\},$$

$$\boldsymbol{F}\text{-}\mathbf{Err_{io}}\boldsymbol{\mathcal{C}} := \{(L,\mu) \mid \exists S \in \mathcal{C}\ \exists_{io} n\ :\ \mathrm{Prob}_\mu[x \in (L \triangle S) \mid x \in \Sigma^n] \ \leq\ F(n)\}$$

where $\boldsymbol{A \,\triangle\, B} := (A \setminus B) \cup (B \setminus A)$ *as the symmetric difference of sets A and B.*

Figure 1 illustrates the error behavior of languages $S_1, S_2, S_3 \in \mathcal{C}$ with respect to a given language L. S_1 provides smaller error probability than F for all inputs. Hence, S_1 proves that $L \in F\text{-Err}\,\mathcal{C}$. The error probability of S_2 will infinitely often fall below the error bound F. If no language in $\mathcal{C}$ with the behavior of

S_1 and S_2 exists, L cannot be approximated by any language in $\mathcal{C}$. It follows that $L \notin F\text{-Err}_{\text{io}}\,\mathcal{C}$. Figure 2 illustrates the error probability of S w.r.t. L: $L \in F_1\text{-Err}\,\{S\}$ and $L \in F_2\text{-Err}_{\text{io}}\,\{S\}$, but $L \notin F_2\text{-Err}\,\{S\}$ and $L \notin F_3\text{-Err}_{\text{io}}\,\{S\}$.

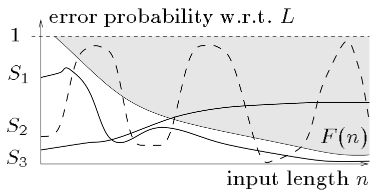

Fig. 1. The error probability of languages S_1, S_2, and S_3 with respect to L for increasing input length n.

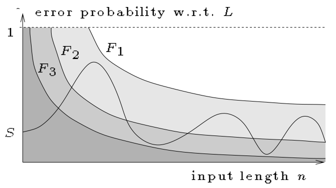

Fig. 2. F_3 gives a lower bound of the error probability of a language S with respect to the io-measure. The error bound F_1 gives an upper bound for both classes.

Using definition 1 we can generalize the classes Nearly-$\mathcal{BPP}$ (introduced by Yamakami [Yam1 96,Yam2 96]) and Nearly-$\mathcal{P}$ by **Nearly-$\mathcal{C}$** $:= n^{-\omega(1)}\text{-Err}\,\mathcal{C}$ for an arbitrary class $\mathcal{C}$. Nearly-$\mathcal{C}$ represent complexity classes with a very low error probability.

Computational classes do not only describe the feasibility of problems, sometimes they are used to ensure intractability. An important application is cryptography, where one wants to prove the security of encryption algorithms, interactive protocols, digital signature schemes, etc. The security is often based on *intractability assumptions* for certain problems, e.g. factoring of large numbers or the computation of quadratic residuosity. Problems are called intractable if every polynomial time bounded algorithm outputs errors for a sufficient high number of instances.

An adequate measure of lower bounds turns out to be $F\text{-Err}_{\text{io}}\,\mathcal{C}$, which generalizes existing models of cryptographic intractability, e.g.

Definition 2 *[GMR 88] A function f is* ***GMR-intractable*** *for a probability distribution μ if for all probabilistic polynomial time bounded algorithms A it holds $\forall c > 0\ \forall_{ae} k\ :\ \text{Prob}_\mu[A(x) = f(x) \mid x \in \Sigma^k] \leq \frac{1}{k^c}$.*

Let $\mathcal{FBPP}$ be the functional extension of $\mathcal{BPP}$, or more precisely: $f \in \mathcal{FBPP}$ iff there exists a polynomial time bounded probabilistic Turing machine M such that $\text{Prob}[M(x) \neq f(x)] \leq \frac{1}{3}$. Note that the error of $\frac{1}{3}$ can be decreased to any polynomial, without loosing the polynomial time behavior of M. To classify GMR-intractable functions by error complexity classes we take an appropriate generalization of $F\text{-Err}_{\text{io}}\,\mathcal{C}$ for functional classes $\mathcal{FC}$, i.e. $(f,\mu) \in F\text{-Err}_{\text{io}}\,\mathcal{FC}$ iff

$$\exists g \in \mathcal{FC}\ \ \exists_{io} n\ :\ \ \text{Prob}_\mu[f(x) \neq g(x) \mid x \in \Sigma^n]\ \leq\ F(n)\ .$$

Proposition 1 *GMR-intractable functions are not in* $\left(1 - n^{-\Omega(1)}\right)\text{-Err}_{\text{io}}\,\mathcal{FBPP}$.

The intractability assumption of [GoMi 82] and the *hard-core* sets of [Imp1 95] refer to non-uniform Boolean circuits. These classes can analogously be expressed by using error complexity classes.

In the rest of this paper we concentrate our considerations on complexity classes of languages and the uniform distribution μ_{uni} as underlying probability distribution, where for all $n \in \mathbb{N}$ and $x, y \in \Sigma^n$ holds $\mu_{\text{uni}}(x) = \mu_{\text{uni}}(y) > 0$. For the sake of readability we omit the distribution:

$$\boldsymbol{L \in F}\textbf{-Err}\boldsymbol{\mathcal{C}} \;:\Leftrightarrow\; (L, \mu_{\text{uni}}) \in F\text{-Err}\,\mathcal{C}\ ,$$

$$\boldsymbol{L \in F}\textbf{-Err}_{\textbf{io}}\boldsymbol{\mathcal{C}} \;:\Leftrightarrow\; (L, \mu_{\text{uni}}) \in F\text{-Err}_{\text{io}}\,\mathcal{C}\ .$$

In this paper we show a first classification of suitable error bounds and extend these lower bound results to time hierarchies. In section 4 we discuss in detail the error complexity of the halting problem and the Kolmogorov complexity problem. Non-recursiveness does not imply high error bounds in general. For the halting problem the Gödel-enumeration of programs has to be examined under new aspects. For some enumerations there are algorithms computing the halting problem within small error probability. We show in the following that even a standard enumeration yields a high lower error bound. In the case of Kolmogorov-complexity the lower bound is even higher and worst possible: We give a constant bound independent from the encoding.

2 Notations

For a predicate $P(n)$ let $\boldsymbol{\forall_{ae} n :\ P(n)}$ be equivalent to $\exists n_0 \forall n \geq n_0 :\ P(n)$ and $\boldsymbol{\exists_{io} n :\ P(n)}$ to $\forall n_0 \exists n \geq n_0 :\ P(n)$. Further, define $f(n) \boldsymbol{\leq_{ae}} g(n)$ as $\forall_{ae} n :\ f(n) \leq g(n)$ and $f(n) \boldsymbol{\leq_{io}} g(n)$ as $\exists_{io} n :\ f(n) \leq g(n)$.

We consider strings over an at least binary alphabet $\boldsymbol{\Sigma}$, where $\boldsymbol{\lambda}$ denotes the empty word. Define $\boldsymbol{\Sigma^{\leq n}} := \bigcup_{i \leq n} \Sigma^i$. Further, we use the lexicographical order function $\text{ord} : \Sigma^* \mapsto \mathbb{N}$ as straightforward isomorphism and its reverse function $\text{str} : \mathbb{N} \mapsto \Sigma^*$. $\mathcal{RE}$ and $\mathcal{REC}$ define the sets of all partial recursive, resp. recursive predicates. For a partial function f the domain is called $\textbf{dom}\boldsymbol{(f)}$. We use $f(x) = \perp$ to denote $x \notin \text{dom}(f)$. Furthermore, let $\boldsymbol{L_0, L_1, L_2, \ldots}$ be an enumeration of the languages in $\mathcal{RE}$ over a given alphabet Σ. For easier notation we present a language L_i resp. $L_i[a, b]$ by its **characteristic string** $L_i[a, b] := L_i(a) \cdots L_i(b)$, where $L_i(\text{ord}(w)) = 1$ if $w \in L_i$ and 0 otherwise.

For a partial recursive function φ and for all $x, y \in \Sigma^*$ define the **relative Kolmogorov complexity** as $\boldsymbol{C_\varphi(x|y)} := \min\{|p| :\ \varphi(p, y) = x\}$. A programming system φ is called **universal** if for all partial recursive functions f holds $\forall x, y \in \Sigma^* :\ C_\varphi(x|y) \leq C_f(x|y) + O(1)$. Fixing such a universal programming system φ we define $\boldsymbol{C_\varphi(x)} := C_\varphi(x|\lambda)$ as the (absolute) **Kolmogorov complexity** of x.

3 The Bounds of Error Complexity Classes

When the error probability tends to 1, the error complexity becomes meaningless. But for the io-error classes the corresponding probability is $\frac{1}{2}$.

Proposition 2 *Let $\mathcal{C}, \mathcal{C}'$ be complexity classes, where $\mathcal{C}$ is closed under finite variation and $\mathcal{C}'$ contains at least two complementary languages. Then for all* *functions* $z(n) =_{ae} 0$ *and* $z'(n) =_{io} 0$ *it holds that*

$$2^{\Sigma^*} = \left(1 - \frac{z(n)}{|\Sigma|^n}\right)\text{-Err}\,\mathcal{C} \qquad \textit{and} \qquad 2^{\Sigma^*} = \left(\frac{1}{2} - \frac{z'(n)}{|\Sigma|^n}\right)\text{-Err}_{\text{io}}\,\mathcal{C}' \ .$$

This only holds for decision problems—the situation for functional classes is very different. Further note that this proposition holds for all classes $\mathcal{C}$ covering the set of regular languages. Consequently, all languages (even the non-recursive languages) can be computed by a finite automaton within these error probabilities. The first upper error bound of this proposition is sharp: Using a delayed diagonalization technique we can construct a language L that cannot be computed with an arbitrary Turing machine within an error probability $1 - \frac{e(n)}{|\Sigma|^n}$ if $e(n) \geq_{io} 1$.

Theorem 1 *There exists a language, such that for all funct.* $e(n) \geq_{io} 1$ *it holds*

$$L \quad \notin \quad \left(1 - \frac{e(n)}{|\Sigma^n|}\right)\text{-Err}\,\mathcal{RE} \ .$$

To prove that the upper bound of the io-error complexity measure is tight in the limit we will use the following technical lemma dealing with the **Hamming distance** $|x - y|$ of two binary strings $x, y \in \{0,1\}^n$.

Lemma 1 *Let* $\boldsymbol{m(k,n)} := \sum_{i=0}^{k} \binom{n}{i}$ *then it holds*

1. *Let* $x_1, \dots x_k \in \{0,1\}^\ell$ *and* $d \in \mathbb{N}$ *with* $k \cdot m(d,\ell) < 2^\ell$. *There exists* $y \in \{0,1\}^\ell$ *such that* $\min_{i\in\{1,\dots,k\}}(|x_i - y|) \geq d$.
2. *For* $\alpha < \frac{1}{2}$ *and* $\alpha \cdot n \in \mathbb{N}$ *it holds that* $m(\alpha \cdot n, n) < \binom{n}{\alpha \cdot n} \cdot \frac{1-\alpha}{1-2\alpha}$.
3. *For* $f \in \omega(\sqrt{n})$ *there exists* $g \in \omega(1) \ : \ g(n) \cdot m\left(n/2 - f(n), n\right) <_{ae} 2^n$.

One may wonder whether high error complexity implies high Kolmogorov complexity. At least the contrary is true.

Lemma 2

$$L \in \frac{f(n)}{|\Sigma|^n}\text{-Err}\,\mathcal{REC} \quad \Longrightarrow \quad \exists c\ \forall n \ : \ C(L \cap \Sigma^n) \leq \log m(f(n), |\Sigma^n|) + c$$

$$L \in \frac{f(n)}{\Sigma^n}\text{-Err}_{\text{io}}\,\mathcal{REC} \quad \Longrightarrow \quad \exists c\ \exists_{io} n \ : \ C(L \cap \Sigma^n) \leq \log m(f(n), |\Sigma^n|) + c$$

In [GGH 93] a similar result is presented. They relate average time complexity and time-bounded Kolmogorov complexity.

The Kolmogorov complexity of enumerable sets is low, since for all enumerable sets L it holds $\forall n \ : C(L \cap \Sigma^n) \leq n + O(1)$. For an excellent survey over this field see [LiVi 97]. We show an explicit construction of a diagonal language with low Kolmogorov complexity (comparable to an enumerable set), giving a strict lower bound for the io-error complexity. This language L will be constructed by using the Hamming distance between k sublanguages $L_0[i, i+\ell], \dots, L_k[i, i+\ell]$ of length ℓ for increasing k and ℓ. We will show that L cannot be approximated by any Turing-machine within an io-error of c for any constant $c < 1/2$.

Theorem 2 *For any* $\varepsilon \in \omega(1)$ *there exists* $L \notin \left(\frac{1}{2} - \frac{1}{o(|\Sigma|^{n/2})}\right)\text{-Err}_{\text{io}}\,\mathcal{RE}$ *such that* $\forall n \ : \ C(L \cap \Sigma^n) \leq n + \varepsilon(n)$.

Proof: For a function $f \in \omega(\sqrt{m})$, let $\alpha(m) := \frac{1}{2} - \frac{f(m)}{m}$, $g(m) := \frac{f(m)\cdot\sqrt{8\pi m}}{m-2f(m)}$, and $\gamma(x) := \min\{\ \ell \in \mathbb{N} \mid g(|\Sigma|^{\ell}) \geq x\ \}$.

We construct a language L which cannot be approximated by partial recursive languages $L_0, L_1, \ldots$ within an io-error probability of $\alpha(|\Sigma|^n)$ as follows: Define $b_i := |\Sigma^{<i}|$, $k_i := |\Sigma^i|$ and choose a sublanguage $S_{\ell,i}$ *lexicographically minimal* such that for all $j \leq \ell$ holds $|L_j\,[b_i, b_{i+1}-1] - S_{\ell,i}| \geq \alpha(k) \cdot k_i$. From Lemma 1 we can conclude that for all $\ell \in \mathbb{N}$ and $c \geq 1$ such a language $S_{\ell,\delta(\ell)+c}$ always exists. Finally, we define the language L as the concatenation of the sublanguages $S_{\ell_i,i}$ for $i = 0, 1, 2, \ldots$ and $\ell_i := \lfloor g(|\Sigma|^i) \rfloor$. From the definition of $S_{\ell_i,i}$ it follows, that for each language L_i, there exists an index j, such that for all $n \geq j$ the Hamming distance of $L_i \cap \Sigma^n$ and $L \cap \Sigma^n$ is at least $\alpha(|\Sigma|^n) \cdot |\Sigma|^n$. That means, L_i cannot predict the language L restricted to words of length n within an error probability smaller than $\alpha(|\Sigma|^n) = \frac{1}{2} - \frac{1}{o(|\Sigma|^{n/2})}$.

On the other hand the sequence $L_0\,[b_n,\ b_{n+1}-1] \ldots L_{g(n)}\,[b_n,\ b_{n+1}-1]$ can be reconstructed if $g(n)$, n, and the number of elements of the corresponding sublanguages are known. Thus, it has a Kolmogorov complexity of at most $O(\log g(n)) + \log(g(n) \cdot |\Sigma^n|) + c_1$ for constant c_1. Using this sequence we can easily construct $L \cap \Sigma^n$. Hence, the Kolmogorov complexity of $L \cap \Sigma^n$ is bounded by $n + \varepsilon(n)$ for an arbitrarily small $\varepsilon \in \omega(1)$ if f is chosen appropriately. ∎

If we restrict ourselves to computational complexity classes we can apply the results shown so far also to classes specified by time bounded Turing machines. Let $\mathbf{DTime}(\boldsymbol{T})$ resp. $\mathbf{DTime}_{\boldsymbol{k}}(\boldsymbol{T})$ be the class of all languages which can be accepted by a T-time bounded deterministic (k-tape) Turing machine. We call a function f **$\boldsymbol{T}$-time $\boldsymbol{k}$-tape computable**, if $f \in \mathcal{F}\mathbf{DTime}_k(T)$ and it is called **time constructible**, if $T \in \mathcal{F}\mathbf{DTime}_2(T)$. In [CaSe 99,ReSc 96] tight average time hierarchies are presented for very carefully defined average time classes. We state corresponding hierarchies for both error-classes following from Theorem 1 and 2.

Corollary 1 *Let T_1, T_2 be two time-constructible functions with $T_1 \in \omega(T_2)$ and $f \geq_{io} 1$, $f' \in o(\sqrt{N})$ T_1-time computable functions. Then for $k \geq 2$ there exists a function $\delta \in \omega(1)$ with $\delta \cdot T_2 \in o(T_1)$ such that it holds*

$$\begin{aligned}
\mathrm{DTime}_k(T_1) &\not\subseteq \left(1 - \tfrac{f(n)}{|\Sigma^n|}\right)\text{-Err DTime}_k(T_2)\ , \\
\mathrm{DTime}_k(T_1) &\not\subseteq \left(\tfrac{1}{2} - \tfrac{1}{f(|\Sigma|^{\delta(n)})}\right)\text{-Err}_{\mathrm{io}}\ \mathrm{DTime}_k(T_2)\ .
\end{aligned}$$

Hence, there are languages computable in time T which cannot be accepted by a Turing machine with asymptotically slower running time within an error significantly smaller than $\frac{1}{2}$. Of course, these results can also be transfered to other computational resources like space or reversals. To transfer these results to DTime(T), an additional factor of $\log T$ for the tape reduction has to be taken into account.

4 Partial Recursive Functions with High Error Bounds

In the last section we showed that there are languages which cannot be approximated by any partial recursive language within an io-error bound significantly smaller than 1/2. To identify some well known languages which cannot be approximated by a partial recursive language within a (nearly) constant io-error fraction we consider the Halting Problem and the Kolmogorov complexity. Note that their complements are not partial recursive.

We will show that both problems cannot be solved by a recursive function within an io-error smaller than a constant, with the constant depending on the chosen programming system. That means that any algorithm, like a universal program checker, claiming to solve one of these problems within a neglectable small error, fails.

4.1 Lower Bounds for the Halting Problem

The halting problem occurs for various models of computation. Obviously, the error complexity depends on the chosen programming system. To get a general approach, we follow the notation of [Smit 94]:

Definition 3 *A* ***programming system*** $\boldsymbol{\varphi}$ *is a sequence* $\varphi_0, \varphi_1, \varphi_2, \ldots$ *of all partial recursive functions such that there exists a universal program* $u \in \mathbb{N}$ *with* $\varphi_u(\langle i, x\rangle) = \varphi_i(x)$ *for a bijective function* $\langle\cdot,\cdot\rangle : \boldsymbol{\Sigma}^* \times \boldsymbol{\Sigma}^* \longrightarrow \boldsymbol{\Sigma}^*$ *called* ***pairing function****. The* ***halting problem*** $\boldsymbol{H_\varphi}$ *for a programming system* φ *is defined as: Given a pair* $\langle i, x\rangle$*, decide whether* $x \in \mathrm{dom}(\varphi_i(x))$*.*

The programming system highly influences the error complexity of the halting problem. Consider for example a programming system for a binary alphabet where $\psi_{2^i} \equiv \varphi_i$ and ψ_j describes the identity function for all $j \neq 2^i$. Of course this anomalous programming system allows a program to compute the halting problem within exponential small error probability. To restrict the programming systems we define:

Definition 4 *The* ***repetition rate*** *of* ***domain equivalent partial functions*** $\boldsymbol{RD_{\varphi,i}}$ *is defined as* $\boldsymbol{RD_{\varphi,i}(n)} := \mathrm{Prob}[\mathrm{dom}(\varphi_x) = \mathrm{dom}(\varphi_i) \mid x \in \Sigma^n]$*. A programming system* φ *is* ***dense****, iff* $\forall i\ \exists c > 0 \ :\ RD_{\varphi,i}(n) \ \geq_{ae}\ c$*.*

Note that most of *real world* programming systems, like PASCAL, are dense.

The second parameter directly influencing the error complexity of the halting problem is the pairing function of the universal program. We can change the situation considerably, if the pairing function is chosen appropriately. Then we can achieve the highest possible error complexity using a diagonalization.

Theorem 3 *There exists a pairing function such that for all programming systems* φ *and for any function* $f <_{ae} 1$ *it holds* $\quad H_\varphi \ \notin\ f\text{-}\mathrm{Err}\,\mathcal{REC}$

This only shows that an artificial pairing function can cause high error complexity. To derive more general results we define the notion of pair-fairness.

Definition 5 *We call a pairing function* **pair-fair**, *if for sets* $X, Y \subseteq \Sigma^*$ *with* $\exists c_1 > 0\ \forall n : \frac{|X \cap \Sigma^n|}{|\Sigma^n|} \geq c_1$ *and* $\exists \ell_1, \ell_2 \in \mathbb{N} : Y = \{w \mid \mathrm{ord}(w) \equiv \ell_1 \pmod{\ell_2}\}$ *it holds* $\exists c_2\ \forall_{ae} n\ :\ \ \mathrm{Prob}[x \in X \wedge y \in Y \mid \langle x, y\rangle \in \Sigma^n] \geq c_2$.

Proposition 3 *The standard pairing* $\langle x, y\rangle = x + \frac{(x+y)(x+y+1)}{2}$ *is pair-fair.*

In the following we restrict our considerations to fair pairing functions $\langle\cdot,\cdot\rangle$. Before we can prove a lower bound of the error complexity of the halting problem, we have to show the following technical lemma.

Lemma 3 *For a pair-fair function* $\langle\cdot,\cdot\rangle$ *and a pairing function* $\langle\!\langle\cdot,\cdot\rangle\!\rangle$ *it holds* $\forall i\ \exists j\ \forall x\ \forall y :\ \mathrm{dom}(\varphi_x) = \mathrm{dom}(\varphi_j) \implies \varphi_i(\langle x, \langle\!\langle x, y\rangle\!\rangle\rangle) \neq H_\varphi(\langle x, \langle\!\langle x, y\rangle\!\rangle\rangle)$.

Choosing $\langle\!\langle x, y\rangle\!\rangle := 2^x \cdot (2y+1) - 1$, we can conclude from Lemma 3:

Theorem 4 *For any dense programming system* φ *it holds*

$$\forall M\ \exists \alpha > 0\ \forall_{ae} n\ :\ \ \mathrm{Prob}[M(x) \neq H_\varphi(x) \mid x \in \Sigma^n] \geq \alpha\ .$$

This means that every heuristic that claims to solve the halting problem makes at least a constant fraction of errors.

Corollary 2 *For any dense progr. system* φ *and any function* $f \in \omega(1)$, *it holds* $H_\varphi \notin \frac{1}{f}\text{-}\mathrm{Err}_{\mathrm{io}}\,\mathcal{REC}$.

The question whether or not there exists a constant lower bound for the io-error complexity of halting is still open. Perhaps the trivial constant upper bound can be improved by showing that for a sequence of Turing machines the error complexity tends to zero in the limit. The last corollary implies that $H_\varphi \notin$ Nearly-$\mathcal{REC}$. Thus, even an improved upper bound would not be helpful for practical issues.

4.2 Lower Error Bounds for Kolmogorov Complexity

Another well known non-recursive problem is Kolmogorov complexity. One of the fundamental results of Kolmogorov complexity theory is the proof of the existence of a *universal programming system*. Such a programming system is not necessarily dense, although many programming systems provide both properties. A sufficient condition for both features is the capability of the universal program to store a fixed input parameter one-to-one into the index string, that means there exists a function $s : \Sigma^* \to \Sigma^*$ such that for all x, y it holds $\varphi_u(\langle x, y\rangle) = \varphi_{s(x)}(y)$ and $|s(x)| = |x| + O(1)$. This observation implies a trivial upper bound for the Kolmogorov complexity of x.
We consider the following decision problem based on the Kolmogorov complexity for a classification of its io-error complexity.

Definition 6 *For a function* $f : \mathbb{N} \to \mathbb{N}$ *define* $\boldsymbol{C_{\leq f}}$ *as the set of all inputs* x *with Kolmogorov complexity smaller than* $f(|x|)$, *i.e.* $\boldsymbol{C_{\leq f}} := \{x \in \Sigma^* \mid \mathrm{C}(x) \leq f(|x|)\}$. *For a constant* c *we define the function* $\boldsymbol{\kappa_c} : \mathbb{N} \to [0,1]$ *as* $\boldsymbol{\kappa_c(n)} := \mathrm{Prob}[\mathrm{C}(x) \leq n - c \mid x \in \Sigma^n]$.

In general, the functions C and κ_c are not recursive. But at least $C_{\leq f}$ is recursively enumerable. In the following we will investigate the size of $\mathrm{C}_{\leq n-c}$ and show a linear lower and upper bound.

Lemma 4 *For any constant $c \geq 1$ there exist constants $k_1, k_2 > 0$ such that $k_1 \leq_{ae} \kappa_c \leq_{ae} 1 - k_2$.*

It is well known that for small recursive functions $f, g \leq \log n$ with $f \in \Omega(g)$ and $g \in \omega(1)$ the set $\mathrm{C}_{\leq f}$ is partially recursive. Furthermore, no infinite recursive enumerable set A is completely included in $\overline{\mathrm{C}_{\leq f}}$, i.e. $A \cap \mathrm{C}_{\leq f} \neq \emptyset$. The following Lemma substantiates the size of this non-empty set $A \cap \mathrm{C}_{\leq f}$ for $f(n) = n - c$.

Lemma 5 *Let $A \in \mathcal{RE}$ such that $|A \cap \Sigma^n| \geq_{ae} c_1 \cdot |\Sigma^n|$ for constant $c_1 > 0$. Then it holds $\forall c_2 > 0\ \exists c_3 > 0\ :\ \ \mathrm{Prob}[\mathrm{C}(x)\ \leq\ n - c_2 \mid x \in A \cap \Sigma^n]\ \geq_{ae}\ c_3$.*

Using these lemmas the following lower bound can be shown.

Theorem 5 *For any $c \geq 1$ there exists a constant $\alpha < 1$ such that $\mathrm{C}_{\leq n-c} \notin \alpha\text{-}\mathrm{Err}_{\mathrm{io}}\,\mathcal{REC}$.*

Proof: For a machine M define $K_n := \Sigma^n \cap \mathrm{C}_{\leq n-c}$, $A_n := \{x \in \Sigma^n \mid M(x) = 0\}$, and $F_n := \{x \in \Sigma^n \mid M(x) \neq C_{\leq n-c}(x)\}$ for all $n \in \mathbb{N}$.
Note that $\Sigma^n \setminus (A_n \bigtriangleup K_n) \subseteq F_n$ and $A_n \cap K_n \subseteq F_n$. From Lemma 4 we can conclude that $k_1 \cdot |\Sigma^n| \leq_{ae} |K_n| \leq_{ae} (1-k_2) \cdot |\Sigma^n|$ and therefore either $c_1 \cdot |\Sigma^n| \leq_{ae} |A_n|$ or $|A_n \bigtriangleup K_n| \leq_{ae} (1 - c_2) \cdot |\Sigma^n|$ for some constants $k_1, k_2, c_1, c_2 > 0$. Using Lemma 5 it follows that for a constant $c_3 > 0\ \ |A_n \cap K_n| \geq_{ae} c_3 \cdot |\Sigma^n|$ or $|A_n \bigtriangleup K_n| \leq_{ae} (1 - c_2) \cdot |\Sigma^n|$. Finally, we can conclude that for some constant $c_4 > 0$ it holds $|F_n| \geq_{ae} c_5 \cdot |\Sigma^n|$. ∎

It follows that there exists a fixed constant α such that no matter which algorithm tries to compute $\mathrm{C}_{\leq n-c}$ it fails for at least a fraction of α of all inputs.

5 Discussion

One might expect that the concept of immune sets and complexity cores are suitable for showing lower bounds. Recall that a set S is called *$\mathcal{C}$-immune* if it has no infinitive subset in $\mathcal{C}$. S is called *$\mathcal{C}$-bi-immune* if S and $\overline{S}$ are immune for $\mathcal{C}$. A recursive set X is called a *complexity core* of S if for every algorithm M recognizing S and every polynomial p, the running time of M on x exceeds $p(|x|)$ on all but finitely many $x \in X$.

Orponen and Schöning [ScOr 84] observed that a set $S \notin \mathcal{P}$ is bi-immune for $\mathcal{P}$ iff Σ^* is a complexity core for S. But this does not imply reasonable lower bounds for the error complexity, since a precondition for complexity cores is the correct computation of an algorithm. Since bi-immune sets may be very sparse, even the trivial language $\emptyset$ gives a low error bound. Thus, the knowledge of a bi-immune set S for $\mathcal{C}$ does not result in a high error complexity. It is an open problem how density for bi-immune sets has to be defined such that reasonable results for the error complexity can be achieved.

However, we can show that the existence of a immune set of a class $\mathcal{C}$ corresponds to a small error bound separation:

Theorem 6 *Let $\mathcal{C}$ be closed under finite variation and $\emptyset \in \mathcal{C}$. There exists an $\mathcal{C}$-immune set iff $\mathcal{C} \neq \frac{1}{|\Sigma^n|}\text{-Err}\,\mathcal{C} \cap \mathcal{RE}$.*

Since some elementary closure properties of $\mathcal{C}$ guarantee the existence of a $\mathcal{C}$-immune set [BoDu 87], this restriction is not severe. On the other hand in [Yesh 83] it is shown that the structural property of *conjunctively-self-reducibility* suffices to overcome all erroneous outputs.

As shown in section 4 the Kolmogorov complexity problem, i.e. the question to decide whether a string can be compressed more than a constant, cannot be computed by any machine within a smaller error probability than a constant. It is notable that this error probability is independent from the machine. Both, the halting and the Kolmogorov problem are not in Nearly-$\mathcal{BPP}$ and remain intractable with respect to the set of recursive predicates.

Because of their strong relationship to the halting problem some other problems – like program verification or virus-program detection – are not recursive in the general case, too. So, it seems that the lower bounds proved so far influence the error bounds of these problems. The exact classification is still an open problem.

Acknowledgment We would like to thank Karin Genther, Rüdiger Reischuk, Arfst Nickelsen, Gerhard Buntrock, Hanno Lefmann, and Stephan Weis for helpful suggestions, critics and fruitful discussions. Furthermore, we thank several unknown referees and the members of the program-comitee for their suggestions, comments and pointers to the literature.

References

[BCG 92] S. Ben-David, B. Chor, O. Goldreich, M. Luby, *On the Theory of Average Case Complexity*, Journal of Computer and System Sciences, Vol. 44, 1992, 193-219.

[BoDu 87] R. Book, D. Du, *The Existence and Density of Generalized Complexity Cores*, Journal of the ACM, Vol. 34, No. 3, 1987, 718-730.

[CaSe 99] J-Y. Cai and A. Selman, *Fine separation of average time complexity classes*, SIAM Journal on Computing, Vol. 28(4), 1999, 1310–1325.

[DaPu 60] Martin Davis, Hillary Putnam, *A Computing Procedure for Quantification Theory*, Journal of the ACM, 1960, 201-215.

[Fagi 92] B. Fagin, *Fast Addition for Large Integers*, IEEE Tr. Comput. Vol. 41, 1992, 1069-1077.

[GeHo 94] P. Gemmel, M. Horchol *Tight Bounds on Expected Time to Add Correctly and Add Mostly Correctly*, Inform. Proc. Letters, 1994, 77-83.

[GGH 93] M. Goldmann, P. Grape and J. Håstad. *On average time hierarchies*, Information Processing Letters, Vol. 49(1), 1994, 15-20.

[GoMi 82] S. Goldwasser, S. Micali, *Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information*, Proc. 14th Annual ACM Symposium on Theory of Computing, 1982, 365-377.

[GMR 88] S. Goldwasser, S. Micali, R. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks*, SIAM J. Comput. Vol.17, No. 2, 1988, 281-308.

[Gure 91] Y. Gurevich, *Average Case Completeness*, Journal of Computer and System Sciences, Vol. 42, 1991, 346-398.

[Hoos 98] H. Hoos, *Stochastic Local Search - Methods, Models, Applications*, Dissertation, Technical University Darmstadt , 1998.

[Imp1 95] R. Impagliazzo, *Hard-core distributions for somewhat hard problems*, In 36th Annual Symposium on Foundations of Computer Science, 1995, 538-545.

[Imp2 95] R. Impagliazzo, *A personal view of average-case complexity*, In Proceedings of the Tenth Annual Structure in Complexity Theory Conference, 134-147, 1995.

[ImWi 98] R. Impagliazzo, A. Wigderson, *Randomness vs. Time: De-randomization under a uniform assumption*, Proc. 39th Symposium on Foundations of Computer Science, 1998, 734–743.

[Levi 86] Leonid Levin, *Average Case Complete Problems*, SIAM Journal on Computing, Vol. 15, 1986, 285-286.

[LiVi 97] M. Li, P. Vitáni, *An Introduction to Kolmogorov Complexity and its Application*, Springer, 1997.

[PuBr 85] P. W. Purdom, C. A. Brown, *The Pure Literal Rule and Polynomial Average Time*, SIAM J. Comput., 1985, 943-953.

[Reif 83] J. Reif, *Probabilistic Parallel Prefix Computation*, Comp. Math. Applic. 26, 1993, 101-110. Technical Report Havard University, 1983.

[ReSc 96] R. Reischuk, C. Schindelhauer, *An Average Complexity Measure that Yields Tight Hierarchies,* Journal on Computional Complexity, Vol. 6, 1996, 133-173.

[Schi 96] C. Schindelhauer, *Average- und Median-Komplexitätsklassen*, Dissertation, Medizinische Universität Lübeck, 1996.

[ScJa 97] C. Schindelhauer, A. Jakoby, *Computational Error Complexity Classes*, Technical Report A-97-17, Medizinische Universität Lübeck, 1997.

[ScOr 84] U. Schöning, P. Orponen, *The Structure of Polynomial Complexity Cores*, Proc. 11th Symposium MFCS, LNCS 176, 1984, 452-458.

[ScWa 94] R. Schuler, O. Watanabe, *Towards Average-Case Complexity Analysis of NP Optimization Problems*, Proc. 10th Annual IEEE Conference on Structure in Complexity Theory, 1995, 148-159.

[SLM 92] Bart Selman, Hector Levesque, David Mitchell, *Hard and Easy Distributions of SAT Problems*, Proc. 10. Nat. Conf. on Artificial Intelligence, 1992, 440-446.

[Smit 94] C. Smith, *A Recursive Introduction to the Theory of Computation*, Springer, 1994.

[Yam1 96] T. Yamakami, *Average Case Computational Complexity Theory*, Phd. Thesis, Technical Report 307/97, Department of Computer Science, University of Toronto.

[Yam2 96] T. Yamakami, *Polynomial Time Samplable Distributions*, Proc. Mathematical Foundations of Computer Science, 1996, 566-578.

[Yesh 83] Y. Yesha, *On certain polynomial-time truth-table reducibilities of complete sets to sparse sets*, SIAM Journal on Computing, Vol. 12(3), 1983, 411-425.