AN AVERAGE COMPLEXITY MEASURE THAT YIELDS TIGHT HIERARCHIES

R. Reischuk and Chr. Schindelhauer

Abstract.

A new definition is given for the average growth of a function $f : \Sigma^* \to \mathbb{N}$ with respect to a probability measure μ on Σ^*. This allows us to define meaningful average distributional complexity classes for arbitrary time bounds (previously, one could not guarantee arbitrary good precision). It is shown that basically only the ranking of the inputs by decreasing probabilities are of importance.

To compare the average and worst case complexity of problems we study average complexity classes defined by a time bound and a bound on the complexity of possible distributions. Here, the complexity is measured by the time to compute the rank functions of the distributions. We obtain tight and optimal separation results between these average classes. Also the worst case classes can be embedded into this hierarchy. They are shown to be identical to average classes with respect to distributions of exponential complexity.

Key words. worst case complexity, expectation, average complexity, distributional complexity classes, time hierarchies, rank functions, rankability hierarchies.

Subject classifications. 68Q10, 68Q15, 68Q25, 60E05.

1. Introduction and Overview

Levin observed that a sound definition of average complexity and complexity classes is not at all obvious (Levin 86). The classical notion of average-case time complexity of a machine M with respect to given probability distributions

μ_n on inputs x of length n takes the expectation

$$\boldsymbol{E_{\mu_n}(\mathrm{\mathbf{time}}_M)} \;:=\; \sum_{|x|=n} \mu_n(x) \cdot \mathrm{time}_M(x) \;,$$

where $\mathrm{time}_M(x)$ denotes the running time of M on x and μ_n denotes a probability distribution for input length n. The machine M is **μ_n-expected T–time bounded** for a resource bound $T : \mathbb{N} \to \mathbb{N}$, if $\forall n\ E_{\mu_n}(\mathrm{time}_M) \leq T$. That means for all n

$$\sum_{|x|=n} \mu_n(x) \cdot \frac{\mathrm{time}_M(x)}{T(|x|)} \;\leq\; 1 \;.$$

The problem with this definition is that polynomial time simulations of polynomial average time machines can result in superpolynomial average time complexity. For example, consider an algorithm M with running time $\exp(|x|/2)$, if $\exists y : x = yy$ and $|x|$, elsewhere on a binary input string x. Then, for the *local uniform probability distribution* $\mu_{\mathrm{uni},n} \;:=\; \frac{1}{\exp n}$ it holds

$$\forall n\ E_{\mu_{\mathrm{uni},n}}(\mathrm{time}_M) \leq n+1 \;,\ \text{but}\ E_{\mu_{\mathrm{uni},n}}((\mathrm{time}_M)^2) \notin \mathrm{POL} \;.$$

It was resolved by Levin by applying the inverse of T to the fraction, thus requiring

$$\sum_{|x|=n} \mu_n(x) \cdot \frac{T^{-1}(\mathrm{time}_M(x))}{|x|} \;\leq\; 1 \;.$$

This definition does not take into account that the weights of different input length may be very unequal. Thus one considers only distributions μ defined over the whole set of inputs and requires

$$\sum_{x} \mu(x) \cdot \frac{T^{-1}(\mathrm{time}_M(x))}{|x|} \;\leq\; 1 \;.$$

M is then called **(Levin)–μ–average T–time bounded**. For a discussion of this approach see the detailed exposition in Gurevich (91).

From this point we leave the view of Levin, since he considers probabilistic machines. This paper only deals with deterministic Turing machines in the average and their complexity classes which differ from former considered probabilistic average complexity classes.

But no matter which machine model there still remains an unpleasant property in this measure, the influence of the functional growth of $\mu(x)$ on the time bound T. If, for example, one takes the "standard" *uniform probability distribution,* which assigns probability

$$\mu_{\mathrm{uni}}(x) \;:=\; 6/\pi^2 \cdot |x|^{-2} \cdot 2^{-|x|}$$

to a string $x \in \{0,1\}^*$ a machine making n^2 steps on every input of length n would already be average $O(\mathcal{N}^{1+\epsilon})$–time bounded for arbitrary $\epsilon > 0$, where $\mathcal{N}$ denotes the identity function on $\mathbb{N}$. This problem can be resolved to a certain extent (see Gurevich 91), but not completely.

Our first contribution to average complexity will be a new definition of average T–time bounded, which gets rid of this problem. It will allow us to differentiate between arbitrarily close bounds T_1 and T_2 for any $T_1 \leq o(T_2)$. The idea is to bound the complexity of a machine not only with respect to a single probability distribution μ, but with respect to all monotone transformations of μ. A machine which suffices this stricter condition will be called **μ-average T-time bounded.**

At first glance, it seems that this complicates the analysis even more. But we will show that this larger set of conditions is equivalent to a very simple property of the distribution μ, which does not involve probabilities explicitly anymore. The only thing that matters is the ranking of the inputs by μ, that is the sequence of inputs ordered by decreasing probabilities. In practice, one often does not know the values of the distribution exactly, but for each pair of inputs at least one can decide which input is more likely. This way, the whole analysis is greatly simplified.

A **distributional problem** is a pair (L, μ), consisting of a language L and a probability distribution μ. We define *distributional complexity classes* **AvDisTime(T)** containing all pairs (L, μ), for which there exists a DTM accepting L that is μ–average T–time bounded in this generalized sense.

Given a language $L \in \mathrm{DTime}(T)$ and a DTM M for L, it is easy to see that by cycling through all inputs of length n one can find an x, on which M spends the maximal time for inputs of length n. If a probability measure μ gives all its weight for inputs of length n to this x then the average time of M (in the expected sense) with respect to this μ equals the worst case complexity. $\mu(x)$ can be computed in time $O(2^{|x|} \cdot T(|x|))$. Using this idea, Miltersen has shown that allowing exponential time overhead a measure μ can be constructed that is *malign* for all expected T–time bounded machines (Miltersen 93). That means their expected time complexity with respect to this μ is no more than a constant factor smaller than their worst case complexity.

On the other hand, restricting an average analysis to some simple distributions may yield results with little practical value. The satisfiability problem, for example, has been shown quickly solvable for certain symmetric distributions, but the input space generated this way seems to be of not interest for applications in AI (see for example the discussion in Mitchell *et al.* 92). These observation motivate to consider *average complexity classes* **AvTime(T,C)**

consisting of all languages L that can be recognized in μ–average time T for distributions μ of complexity at most C, for certain bounds C. That way, average complexity classes are directly comparable to the standard worst case classes since both contain only languages.

In this paper a notation different from the one in previous research on average complexity will be used because we feel that this new one is more appropriate and natural. There should be a clear distinction between distributional classes, where distributions appear explicitly, and average classes the elements of which are languages in the usual sense. From a complexity theoretic point of view one is more interested in the second kind of classes.

The complexity of a distribution is measured by its **rankability**, that is the effort to compute the ranks of the elements in the input space. Previous approaches have bounded the complexity of distributions using the notion of **computable** and **sampleable** (Levin 86,Gurevich 91,Ben-David *et al.* 92). Preliminary versions of some of these results appeared in Schindelhauer 91 and were presented at the 10. STACS–Conference Reischuk & Schindelhauer 93.

2. Notations

A complexity bound is a function $T : \mathbb{N} \to \mathbb{N}$. Let $\mathcal{N}$ denote the identity function on the natural numbers, i.e. the linear complexity bound. All complexity bounds T considered in this paper are assumed to be monotone increasing, time-constructible and at least as large as $\mathcal{N}$. To simplify the notation, for a constant $\alpha > 0$ the function $T(\alpha\mathcal{N})$ means $n \mapsto T(\lfloor \alpha \cdot n \rfloor)$.

The following sets of complexity bounds will be of special interest:

$$\text{POL} := \bigcup_{k\in\mathbb{N}} O(\mathcal{N}^k) , \quad \text{EXL} := \exp(\Theta(\mathcal{N})) , \quad \text{EEXL} := \exp(\text{EXL}) .$$

If $\mathcal{T}$ is a set of complexity bounds $T \leq \mathcal{T}$ means that for some $T' \in \mathcal{T}$ holds $T \leq T'$. For a complexity bound T, which does not necessarily have to be injective, we define the inverse T^{-1} by

$$T^{-1}(n) := \min\{m \mid T(m) \geq n\} .$$

Useful properties of this definition are

$$T(T^{-1}(n)) \geq n , \quad T^{-1}(T(n)) \leq n , \quad T^{-1}(T(n)+1) > n .$$

Let $M_1, M_2, \ldots$ be an enumeration of all deterministic Turing machines (DTM). In some cases we will also consider nondeterministic machines (NTM). We may assume that all machines have only two work tapes, implying that one can use a universal machine with only a constant factor slowdown. More explicitly, we assume that there is a machine U that can simulate each M_i on any input x in time at most $i \cdot \mathrm{time}_{M_i}(x)$, where $\mathrm{time}_{M_i}(x)$ denotes the number of steps of M_i on input x. $M(x)$ denotes the output that M generates on an input $x \in \Sigma^*$ and $L(M)$ (in case of acceptors) the language accepted by M. Unless otherwise stated, we will always assume that the alphabet Σ has size at least 2. At least for the standard approach using the expectation unary alphabets obviously do not make much sense.

For an ordering of binary strings, $x \leq y$, we refer to the lexicographical ordering. We consider probability measures (density functions) $\mu : \Sigma^* \to [0,1]$ over the input space. μ has to satisfy $\sum_x \mu(x) \leq 1$. $\mathrm{bin} : \mathbb{N} \to \{0,1\}^*$ denotes the standard correspondence between natural numbers and binary strings.

3. Levin's Average Measure

3.1. A Refinement. In the introduction we already discussed the problem how to measure the average time complexity of machines with respect to different probability distributions precisely. Levin's solution essentially can only distinguish between polynomial and superpolynomial growth. The problem with the uniform distribution mentioned above can somehow be diminished, by giving Σ^n instead of a total weight n^{-2} weight proportional to $n^{-1} \cdot \log^{-2} n$ or even better proportional to

$$n^{-1} \cdot \log^{-1} n \cdot \ldots \cdot (\log^{[k-1]} n)^{-1} \cdot (\log^{[k]} n)^{-2}$$

for some k, where $\log^{[k]}$ denotes the k-th iteration of the logarithm function. Still, it can never be completely resolved which can be seen as follows.

Lemma 1. *For a monotone decreasing strictly positive function $u : \Sigma^* \to [0;1]$ and a real-valued function $h : [0,1] \to [0,1]$ with a monotone increasing first derivative h' in $(0,1)$ it holds for $u(x) \neq u(x-1)$*

$$h'(u(x-1)) \;\leq\; \frac{h(u(x)) - h(u(x-1))}{u(x) - u(x-1)} \;\leq\; h'(u(x)) \; .$$

PROOF. Applying the mean value theorem there exists $z_0 \in (0;1)$ such that $h'(z_0) = \frac{h(u(x))-h(u(x-1))}{u(x)-u(x-1)}$. Since h' is monotone increasing, it holds $h'(u(x-1)) \leq h'(z_0) \leq h'(u(x))$. □

LEMMA 2. *For every monotone decreasing strictly positive probability distribution $\mu : \Sigma^* \to [0,1]$ there exists a distribution $\tilde{\mu}$ such that*

$$\lim_{x\to\infty} \frac{\tilde{\mu}(x)}{\mu(x)} = \infty .$$

PROOF. Let $\mu^*(x) := \sum_{z \leq x} \mu(z)$ be the distribution function of μ according to the lexicographical ordering of strings in Σ. Then we define a new probability distribution $\tilde{\mu}$ by the distribution function

$$\tilde{\mu}^*(x) := 1 - \sqrt{1-\mu^*(x)} .$$

Thus the probability distribution is given by $\tilde{\mu}(x) := \tilde{\mu}^*(x) - \tilde{\mu}^*(x-1)$. Then by lemma 1 for $h(x) := 1 - \sqrt{1-x}$ and $u(x) := \mu^*(x)$ we get

$$\frac{\tilde{\mu}(x)}{\mu(x)} \geq \frac{\mu^*(x-1)}{\sqrt{1-\mu^*(x-1)}} .$$

Note that $\frac{\mu^*(x)}{\sqrt{1-\mu^*(x)}}$ tends to infinity. □

Hence, Levin defined only the term *polynomial on the average* as a bound for the time complexity:

DEFINITION 1. *A function $f : \Sigma^* \to \mathbb{N}$ belongs to the class* **Lav(*POL*)** *(Levin average polynomial) with respect to a distribution μ iff for some number k*

$$\sum_x \mu(x) \frac{f(x)^{1/k}}{|x|} < \infty .$$

We will now present a stricter average measure. The idea is to look simultaneously at all distributions $\tilde{\mu}$ which generate the same ordering of the inputs by their probabilities as μ, for example if $\mu(10001) < \mu(11)$ then $\tilde{\mu}(10001) \leq \tilde{\mu}(11)$. μ–average bounded by T is then defined to be $\tilde{\mu}$–average bounded by T in the sense above for all such $\tilde{\mu}$. Thus, only the ranking of the inputs by decreasing probabilities matters. We therefore define for a distribution μ its corresponding rank function rank_μ by

DEFINITION 2.

$$\mathbf{rank}_{\boldsymbol{\mu}}(\boldsymbol{x}) \; := \; \begin{cases} |\{z \in \Sigma^* \mid \mu(z) \geq \mu(x)\}| & \text{if } \mu(x) > 0, \\ \infty & \text{if } \mu(x) = 0, \end{cases}$$

where ∞ *plays the role of a number that is greater than any natural number. A machine computing the rank function is assumed to output a special symbol in the second case. The set*

$$\boldsymbol{X}_{\boldsymbol{\mu}}^{\leq \boldsymbol{l}} \; := \; \{x \mid \mathit{rank}_{\mu}(x) \leq l\}$$

consists of the l *most probable inputs w.r.t to* μ*.* **Uniform distributions** *give identical weights to inputs of the same length. Shorter inputs receive strictly larger probabilities than longer ones. Corresponding to such distributions we have the* **uniform ranking** *of the input space* Σ^* *defined by*

$$\mathbf{rank}_{\mathbf{uni}}(\boldsymbol{x}) \; := \; 1 + |\Sigma| + |\Sigma|^2 + \ldots + |\Sigma|^{|x|} \; .$$

Later, we will also consider special rankings derived from subsets $L \subseteq \Sigma^*$*. For this purpose, let us introduce the notion*

$$\mathbf{rank}_{\boldsymbol{L}}(\boldsymbol{x}) \; := \; \begin{cases} |\{y \in L | y \leq x\}| & \text{if } x \in L, \\ \infty & \text{else.} \end{cases}$$

The set of distributions $\tilde{\mu}$ equivalent to μ can be generated by *monotone transformations* of μ. Figure 3.1 shows a sample of a probability distribution and its rank function.

DEFINITION 3. *A real-valued monotone function* $m : [0,1] \to [0,1]$ *is called a* **monotone transformation** *of the distribution* μ *if* $\sum_x m(\mu(x)) \leq 1$.

DEFINITION 4. *The set* $\mathbf{Av}(\boldsymbol{T})$ *(average* T*) contains all pairs* (f, μ) *consisting of a function* $f : \Sigma^* \to \mathbb{N}$ *and a distribution* μ *such that for all monotone transformations* m *of* μ

$$\sum_x m(\mu(x)) \, \frac{T^{-1}(f(x))}{|x|} \; \leq \; 1 \; .$$

We will show in lemma 3, that this generalization does not destroy polynomial bounds that are increased by a polynomial.

Because of the universal quantifier over all monotone transformations the above condition for $\mathrm{Av}(T)$ is hard to verify. But there exists an equivalent, more practical characterization of $\mathrm{Av}(T)$. Consider the special case of *threshold functions* $\mathtt{thr}_{\boldsymbol{l}} : [0,1] \to [0,1]$ as monotone transformations, where for $l = \mathrm{rank}_{\mu}(x)$ we define $\mathtt{thr}_l(z) := 1/l$ if $z \geq \mu(x)$ and 0 else.

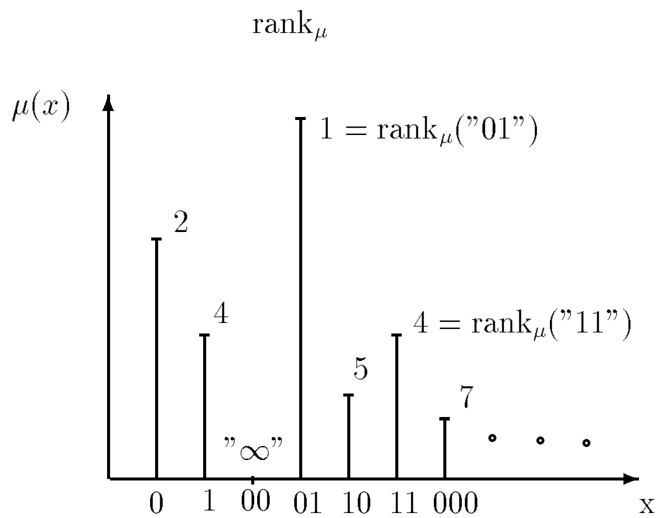

Figure 3.1: The rank function derived from a distribution μ.

PROPOSITION 1.

$$(f,\mu) \in \text{Av}(T) \qquad \Longleftrightarrow \qquad \forall l \quad \sum_x \texttt{thr}_l(\mu(x)) \, \frac{T^{-1}(f(x))}{|x|} \; \leq \; 1 \; .$$

PROOF. "$\Rightarrow$" follows from the definition.
"$\Leftarrow$": Let $x_1, x_2, \ldots$ be an enumeration of all strings s with $\mu(x) \neq 0$ such that $\mu(x_i) \geq \mu(x_{i+1})$. Let $a^n := (a_1, \ldots, a_n)$ where $a_i := T^{-1}(f(x_i))/|x_i|$. Assume $(f,\mu) \notin \text{Av}(T)$. Then there exists an integer n with $\mu(x_n) > \mu(x_{n+1})$ such that for some transformation m it holds $\sum_{i=1}^n m(\mu(x_i)) \cdot a_i > 1$.

To obtain the maximum value of this sum with respect to m we must solve a linear optimization problem. The solution consists of a set of vectors which describe an $(n-r)$–dimensional simplex (r is the number of indices $i < n$ where $\mu(x_i) = \mu(x_{i+1})$). The extremal points are $(\underbrace{1/i, \ldots, 1/i}_{i \text{ times}}, 0, \ldots, 0)$. The solution is the set of points of largest distance from the $(n-1)$-dimensional hyperplane with the normal vector a that contains the origin $(0, \ldots, 0)$. This set contains at least one extremal point of the simplex. Hence, there exists an integer l with $a_1 = a_2 = \cdots = a_l = 1/l$ such that the sum above achieves its maximum. $\square$

As an immediate consequence of this proposition we obtain the following fundamental result, which shows that an average bound can be computed without considering all possible transformations, the only set of interest will be $X_\mu^{\leq l} \;:=\; \{x \mid \text{rank}_\mu(x) \leq l\}$.

COROLLARY 1.

$$(f,\mu) \in \mathrm{Av}(T) \qquad \Longleftrightarrow \qquad \forall l \sum_{x \in X_\mu^{\leq l}} \frac{T^{-1}(f(x))}{|x|} \;\leq\; l \;.$$

PROOF. Observe that for a threshold function $\mathtt{thr}_l$

$$\sum_x \mathtt{thr}_l(\mu(x))\,\frac{T^{-1}(f(x))}{|x|} \;=\; \sum_{x \in X_\mu^{\leq l}} \frac{T^{-1}(f(x))}{l \cdot |x|} \;.$$

So, Corollary 1 follows from proposition 1. □

In the following we will use this simpler characterization to decide membership in $\mathrm{Av}(T)$. Since the condition only involves the ranking of the input space derived from the distribution one also gets

COROLLARY 2. *If μ_1, μ_2 are distributions with $\mathrm{rank}_{\mu_1} = \mathrm{rank}_{\mu_2}$ then*

$$(f,\mu_1) \in \mathrm{Av}(T) \qquad \Longleftrightarrow \qquad (f,\mu_2) \in \mathrm{Av}(T) \;.$$

A rank function ρ represents a whole equivalence class of distributions, namely those μ with $\mathrm{rank}_\mu = \rho$. Therefore, in the following we do not differentiate between pairs (f,μ) containing distributions and pairs (f,ρ) referring to the corresponding rank functions.

The following gives another useful characterization of these sets which helps to simplify estimations.

PROPOSITION 2.

$$(f,\mu) \in \mathrm{Av}(T) \qquad \Longleftrightarrow \qquad \forall l \sum_{x \in X_\mu^{\leq l}} \min\{\alpha_x \mid f(x) \leq T(\alpha_x \cdot |x|)\} \;\;\leq\; l \;.$$

Before we will use this notion to define complexity classes let us investigate closure and transformation properties of these sets of average bounds.

LEMMA 3.
a) For a monotone function $g : \mathcal{N} \to \mathcal{N}$ holds:

$$(f,\mu) \in \mathrm{Av}(T) \qquad \Longrightarrow \qquad (g \circ f,\mu) \in \mathrm{Av}(g \circ T) \;.$$

b) If, in addition, $g > 0$ is strictly increasing also the inverse implications holds:

$$(g \circ f,\mu) \in \mathrm{Av}(g \circ T) \qquad \Longrightarrow \qquad (f,\mu) \in \mathrm{Av}(T) \;.$$

PROOF. a) By proposition 2,

$$\begin{aligned}(f,\mu)\in \mathrm{Av}(T) \quad &\Longrightarrow \quad \forall l \sum_{x\in X_\mu^{\leq l}} \min\{\alpha_x | f(x)\leq T(\alpha_x\cdot|x|)\} \ \leq\ l\\ &\Longrightarrow \quad \forall l \sum_{x\in X_\mu^{\leq l}} \min\{\alpha_x | g(f(x))\leq g(T(\alpha_x\cdot|x|))\} \ \leq\ l\\ &\Longrightarrow \quad (g\circ f,\mu)\in \mathrm{Av}(g\circ T)\ .\end{aligned}$$

b) follows similarly by applying g^{-1}, which is monotone, to $g\circ f$ and $g\circ T$ and noticing that for strictly increasing functions $g^{-1}\circ g = \mathcal{N}$. □

In particular, we get the result that this measure is closed under linear or polynomial transformations.

COROLLARY 3. *For any constants c, c_1, c_2 holds:*

$$\begin{aligned}(f,\mu)\in \mathrm{Av}(T) \quad &\Longrightarrow \quad (c_1 f + c_2,\mu)\in \mathrm{Av}(c_1 T + c_2)\\ &\qquad\quad \text{and} \quad (f^c,\mu)\in \mathrm{Av}(T^c)\\ g\leq \mathit{Pol}(f) \quad \text{and} \quad (f,\mu)\in \mathrm{Av}(T) \quad &\Longrightarrow \quad (g,\mu)\in \mathrm{Av}(\mathrm{Pol}(T))\\ g\leq \mathit{Pol}(f) \quad \text{and} \quad (f,\mu)\in \mathrm{Av}(\mathrm{POL}) \quad &\Longrightarrow \quad (g,\mu)\in \mathrm{Av}(\mathrm{POL})\end{aligned}$$

To estimate sums and products of complexity bounds is a little more complicated. For this purpose, we first will consider the maximum operator. For functions f, g let $\max[f,g]$ denote the function defined by $n\mapsto \max\{f(n),g(n)\}$.

LEMMA 4. *For $0<\beta<1$ holds:*

$$\begin{aligned}(f,\mu)\in \mathrm{Av}(T_f) \quad \text{and} \quad (g,\mu)\in \mathrm{Av}(T_g) \qquad &\Longrightarrow\\ (\max[f,g],\mu) \ \in\ \mathrm{Av}\Big(\max[T_f(\tfrac{\mathcal{N}}{\beta}), T_g(\tfrac{\mathcal{N}}{1-\beta})]\Big)&\end{aligned}$$

PROOF. Let $D \ :=\ \max[T_f(\frac{\mathcal{N}}{\beta}), T_g(\frac{\mathcal{N}}{1-\beta})]$. Then,

$$\begin{aligned}(f,\mu)\in \mathrm{Av}(T_f) \quad &\Longrightarrow \quad \forall l \sum_{x\in X_\mu^{\leq l}} \min\left\{\alpha_x \mid f(x)\leq T_f\left(\frac{\alpha_x}{\beta}\cdot|x|\right)\right\} \ \leq\ l\cdot\beta\\ &\Longrightarrow \quad \forall l \sum_{x\in X_\mu^{\leq l}} \min\{\alpha_x | f(x)\leq D(\alpha_x\cdot|x|)\} \ \leq\ l\cdot\beta\ .\end{aligned}$$

Similarly,

$$(g,\mu)\in \mathrm{Av}(T_g) \ \Longrightarrow\ \forall l \sum_{x\in X_\mu^{\leq l}} \min\{\alpha_x | g(x)\leq D(\alpha_x\cdot|x|)\} \ \leq\ l\cdot(1-\beta)\ .$$

Therefore,

$$\forall l \sum_{x \in X_\mu^{\leq l}} \min\{\alpha_x \mid g(x) \leq D(\alpha_x \cdot |x|) \text{ and } f(x) \leq D(\alpha_x \cdot |x|)\} \;\leq\; l \;,$$

which implies by Proposition 2 that $(\max[f,g],\mu) \in \mathrm{Av}(D)$. □

Hence, for sum and product using $f+g \leq 2 \cdot \max[f,g]$, resp. $f \cdot g \leq \max[f^2, g^2]$ we obtain

Corollary 4.

$$\begin{aligned} (f,\mu) \in \mathrm{Av}(T_f) \quad \text{and} \quad & (g,\mu) \in \mathrm{Av}(T_g) \\ \Longrightarrow \quad & (f+g,\mu) \in \mathrm{Av}(2 \cdot \max[T_f(2 \cdot \mathcal{N}), T_g(2 \cdot \mathcal{N})]) \\ & \text{and } (f \cdot g, \mu) \in \mathrm{Av}(\max[T_f(2 \cdot \mathcal{N})^2, T_g(2 \cdot \mathcal{N})^2]) \;. \end{aligned}$$

3.2. Levin's Measure equals Av(POL) w.r.t Modest Distributions.

Definition 5. *A probability distribution is called modest, if for all x with $\mu(x) \neq 0$*

$$\mu(x) \cdot \mathit{rank}_\mu(x) \cdot \mathit{Pol}(\log(\mathit{rank}_\mu(x))) \;\geq\; 1 \quad \text{and} \quad \mathit{rank}_\mu(x) \;\leq\; \exp(\mathit{Pol}(|x|)) \;.$$

If a probability distribution does not fulfill the first condition then its probabilities may decrease very fast, for example as $\mu(x) = c \cdot 2^{-2^{|x|}}$. Such an extreme probability distribution yields polynomial bounds with respect to Levin's measure even for exponential time functions like $\mathrm{time}(x) = 2^{|x|}$. Such results obviously do not make much sense.

On the other hand the first restriction can always be achieved if we replace the distribution by a appropriately monotone transformed probability distribution. The second inequality requires that the space for describing the rank of an input is bounded by a polynomial.

Theorem 3.1. *For every modest distribution μ and all functions f it holds*

$$(f,\mu) \in \mathrm{Av}(\mathrm{POL}) \quad \Longleftrightarrow \quad (f,\mu) \in \mathrm{Lav}(\mathrm{POL}) \;.$$

Proof. "$\Leftarrow$" by definition of Av (choose the identity function as a monotone transformation).

"$\Rightarrow$" Let $c_1, c_2 \in \mathbb{R}^+$, such that

$$\sum_x \frac{c_1 \cdot f(x)^{c_2}}{|x|} \cdot \mu(x) \quad \leq \quad 1 \;.$$

Since μ is modest, there exist constants $c_3, c_4, c_5 \in \mathbb{R}^+$, such that

$$c_4 \cdot \mu(x) \cdot \mathrm{rank}_\mu(x) \cdot \log^{c_5} \mathrm{rank}_\mu(x) \;\geq\; 1 \quad \text{and} \quad \log \mathrm{rank}_\mu(x) \;\leq\; |x|^{c_3} \;.$$

We will prove, that $(f, \mu) \in \mathrm{Av}\Big((k_1 \cdot \mathcal{N})^{k_2}\Big)$ for $k_1 := \max\Big(2, \frac{2 \cdot c_4}{c_1}\Big)$ and $k_2 := \frac{c_3 \cdot c_5 + 1}{c_2}$. Consider the following sets $I_1 := \{x \mid f(x) \leq |x|^{k_2}\}$ and $I_2 := \{x \mid f(x) > |x|^{k_2}\}$.

1. $f(x) \leq |x|^{k_2}$

   This implies for all $l \in \mathbb{N}$ $\sum_{x \in X_{\overline{\mu}}^{\leq l}} \frac{f(x)^{\frac{1}{k_2}}}{|x|} \;\leq\; l$.

2. $f(x) > |x|^{k_2}$

   Now it holds

$$f(x) \quad \geq \quad |x|^{k_2} \quad \geq \quad \log^{\frac{k_2}{c_3}} \mathrm{rank}_\mu(x) \;.$$

   Thus, for all l and for all x with $\mathrm{rank}_\mu(x) \leq l$

$$\begin{aligned} \frac{f(x)^{\frac{1}{k_2}}}{|x|} \quad &\leq \quad \frac{f(x)^{\frac{c_3 \cdot c_5 + 1}{k_2}}}{f(x)^{\frac{c_3 \cdot c_5}{k_2}} \cdot |x|} \quad \leq \quad \frac{f(x)^{c_2}}{\log^{c_5} \mathrm{rank}_\mu(x) \cdot |x|} \\ \leq \quad \frac{f(x)^{c_2}}{\log^{c_5} \mathrm{rank}_\mu(x) \cdot |x|} \cdot \frac{l}{\mathrm{rank}_\mu(x)} \quad &\leq \quad c_4 \cdot \frac{f(x)^{c_2}}{|x|} \cdot \mu(x) \cdot l \end{aligned}$$

This completes the proof, since

$$\begin{aligned} \sum_{x \in X_{\overline{\mu}}^{\leq l}} \frac{f(x)^{\frac{1}{k_2}}}{k_1 \cdot |x|} \quad &\leq \quad \sum_{x \in X_{\overline{\mu}}^{\leq l} \cap I_1} \frac{f(x)^{\frac{1}{k_2}}}{k_1 \cdot |x|} \;+\; \sum_{x \in X_{\overline{\mu}}^{\leq l} \cap I_2} \frac{f(x)^{\frac{1}{k_2}}}{|x|} \\ &\leq \quad \frac{l}{2} + \sum_{x \in X_{\overline{\mu}}^{\leq l} \cap I_2} \frac{c_1 \cdot f(x)^{c_2}}{2 \cdot |x|} \cdot \mu(x) \cdot l \;\leq\; l \;. \end{aligned}$$

$\square$

When we return from a time bound T to polynomial time bounds we return to the origin: Levin's measure. This gives us the confidence that Av is a reasonable average measure and so we define the following complexity classes.

Definition 6.

$$\begin{aligned} \mathbf{AvDisTime}(\boldsymbol{T}) \;&:=\; \{(L, \mu) \mid \exists M \in DTM: \; L(M) = L \text{ and } \\ &\qquad\qquad (time_M, \mu) \in \mathrm{Av}(T)\} \\ \mathbf{AvDis}\boldsymbol{\mathcal{P}} \;&:=\; \bigcup_{T \in \mathrm{POL}} \mathrm{AvDisTime}(T) \;. \end{aligned}$$

These sets are called **distributional complexity classes**. Their elements are pairs of an algorithmic problem (a language) and a probability distribution. They are formally different from the standard worst case classes. But the ultimate goal is to compare the worst case and the average complexity of problems. Therefore the following definition turns out to be suitable for this purpose. We consider the distributional complexity of languages L with respect to a set C of distributions and require that the average complexity is bounded for all these distributions.

DEFINITION 7. *Let T be a complexity bound and C a set of distributions, resp. rankings. Then*

$$\mathbf{AvTime}(\boldsymbol{T},\boldsymbol{C}) \;:=\; \{L \mid \forall \mu \in C \quad (L,\mu) \in \mathrm{AvDisTime}(T)\} \;.$$

In order to restrict the complexity of the input distributions a natural approach is to put a time limit for computing the corresponding rank functions.

DEFINITION 8. *Let T–**rankable** be the set of all distributions μ, resp. rankings ρ for which there exists a DTM M that on input x computes bin$(\rho(x))$ in time $T(|x|)$.*

Most average complexity classes we will be studying in the following will be of the form

$$\mathrm{AvTime}(T, V\text{–rankable}) \;,$$

where T and V are complexity bounds. Alternative notions for restricting distributions have been defined in Levin 86, Gurevich 91 and Ben-David *et al.* 92.

DEFINITION 9. *A distribution μ belongs to the class T-**computable** if there is a deterministic TM that on input $(x, 1^i)$ outputs the first i bits of the binary expansion of $\mu^*(x) \;:=\; \sum_{z \leq x} \mu(z)$ in time $T(|(x, 1^i)|)$.*
*μ is T-**sampleable** if one can find a probabilistic TM that outputs each string $x \in \Sigma^*$ with probability $\mu(x)$, and this within $T(|x|)$ steps.*

These concepts are not directly comparable because each rank function represents a whole equivalence class of distributions. In a subsequent paper we will discuss the relation between these conditions and the rankability property in more detail.

4. Modification of the Expected Measure

4.1. Bounds for the Expectation Using Rank Functions. The generalization by monotone transformations can also be applied to the classical expected measure.

DEFINITION 10. *The set* **Eav(*T*)** *(expected average T) contains all pairs (f,μ) such that for all monotone transformations m of μ*

$$\sum_x m(\mu(x)) \frac{f(x)}{T(|x|)} \leq 1 .$$

Similar to above this condition can be simplified to

PROPOSITION 3.

$$(f,\mu) \in \mathrm{Eav}(T) \qquad \Longleftrightarrow \qquad \forall l \quad \sum_{x \in X_\mu^{\leq l}} \frac{f(x)}{T(|x|)} \leq l .$$

PROOF. "$\Rightarrow$" by definition.
"$\Leftarrow$": Identical to the proof of proposition 1 and corollary 1 when substituting $\frac{T^{-1}(f(x))}{|x|}$ by $\frac{f(x)}{T(|x|)}$. □

As for the classical expected measure one can easily find examples showing that this measure is not closed under polynomial transformations, too. The complexity classes EavTime(T) are then defined similarly to AvTime(T).

DEFINITION 11. *Let C be a set of distributions, then*

$$\textbf{EavTime}(\boldsymbol{T},\boldsymbol{C}) \ := \ \{L \mid \forall \mu \in C \ \ \exists DTM\ M \ with\ L(M) = L \quad \text{and} \quad (time_M, \mu) \leq \mathrm{Eav}(T)\} .$$

It is not hard to see that the expected run time being bounded by T for every input length, that means $\forall n \quad E_\mu(\mathrm{time}_M) \leq T(n)$, is a stricter condition than membership in Eav(T). A machine contradicting the property $E_{\mu_n}(\mathrm{time}_M) \leq T(n)$ for some n may still be average bounded with respect to the Eav-measure if for smaller input length n' the average is sufficiently smaller than $T(n')$.

There is at least one special case, where both measures define the same complexity class. Consider uniform distributions μ_{uni} with the rank function $\mathrm{rank}_{\mathrm{uni}}(x) := |\Sigma^{\leq |x|}|$ and the local uniform distribution $\mu_{\mathrm{uni},n}(x) := |\Sigma|^{-1}$. Thus, as ranks only values of the form $l = |\Sigma^{\leq |x|}|$ appear. In this case one can prove for an appropriate constant α that only depends on Σ the following equality.

THEOREM 4.1. *For all $T \geq \alpha\mathcal{N}$ holds*

$$\begin{aligned}\mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\}) \;=\; & \{L \mid \exists \text{ a } DTM\ M \text{with } L(M) = L \text{ and} \\ & \qquad \forall n\ E_{\mu_{\mathrm{uni},n}}(time_M) \leq T(n)\ \}\ .\end{aligned}$$

PROOF. "$\subseteq$":
Let $L \in \mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\})$. Hence, there exists a machine M with $L(M) = L$ and $(\mathrm{time}_M, \mu_{\mathrm{uni}}) \in \mathrm{Eav}(T)$. Thus,

$$\begin{aligned}\forall l \sum_{x \in X^{\leq l}_{\mu_{\mathrm{uni}}}} \frac{\mathrm{time}_M(x)}{T(|x|)} \leq l \quad &\Longleftrightarrow \quad \forall n \quad \sum_{|x| \leq n} \frac{\mathrm{time}_M(x)}{T(|x|)} \;\leq\; |\Sigma^{\leq n}| \\ &\Longrightarrow \quad \forall n \quad \sum_{|x| = n} \frac{\mathrm{time}_M(x)}{|\Sigma|^n} \;\leq\; |\Sigma| \cdot T(n)\ .\end{aligned}$$

If M is sped up by the factor $2 \cdot |\Sigma|$ we get a machine M' with $L(M') = L$ and

$$\mathrm{time}_{M'}(x) \;\leq\; \max\left\{\frac{\mathrm{time}_M(x)}{2 \cdot |\Sigma|}, (1+\epsilon)|x|\right\}$$

for some $\epsilon > 0$. If T grows at least as $\alpha\mathcal{N}$ for an appropriate α (for example $\alpha = 8|\Sigma|^2$) then one can show

$$\forall n \qquad \sum_{|x| = n} \frac{\mathrm{time}_{M'}(x)}{|\Sigma|^n} \;\leq\; T(n)\ ,$$

which means that M' is expected T-time bounded.

"$\supseteq$": Let M be a DTM such that $L(M) = L$ and $E_{\mu_{\mathrm{uni}}}(\mathrm{time}_M) \leq T$, which means

$$\begin{aligned}\forall n \quad \sum_{|x|=n} \frac{\mathrm{time}_M(x)}{|\Sigma|^n} \;\leq\; T(n) \quad &\Longleftrightarrow \quad \forall n \quad \sum_{|x|=n} \frac{\mathrm{time}_M(x)}{T(n)} \;\leq\; |\Sigma|^n \\ \Longrightarrow \quad \forall n \quad \sum_{|x| \leq n} \frac{\mathrm{time}_M(x)}{T(n)} \;\leq\; |\Sigma^{\leq n}| \quad &\Longleftrightarrow \quad \forall l \quad \sum_{x \in X^{\leq l}_{\mu_{\mathrm{uni}}}} \frac{\mathrm{time}_M(x)}{T(n)} \;\leq\; l \\ &\Longleftrightarrow \quad (\mathrm{time}_M, \mu_{\mathrm{uni}}) \;\in\; \mathrm{Eav}(T)\ .\end{aligned}$$

$\square$

4.2. EavTime versus AvTime. Let us now compare these two concepts of bounding the average run time. It will be shown that the generalized expected time measure is at least as strict as the average measure. Thus, algorithms with known expected running time can be used to get upper bounds for the average time complexity with respect to the $\mathrm{Av}(T)$ measure.

LEMMA 5. *Let T be a time bound and the function $T/\mathcal{N}$ ($n \mapsto \frac{T(n)}{n}$) be monotone increasing, then*

$$\mathrm{Eav}(T) \subseteq \mathrm{Av}(T(2 \cdot \mathcal{N})) \ .$$

PROOF. Let $(f, \rho) \in \mathrm{Eav}(T)$. Thus,

$$\forall l \qquad \sum_{\rho(x) \leq l} \frac{f(x)}{T(|x|)} \ \leq \ l \ .$$

To prove the claim one has to show

$$\forall l \qquad \sum_{\rho(x) \leq l} \frac{T^{-1}(f(x))}{|x|} \ \leq \ 2 \cdot l \ .$$

Divide the inputs into the two subsets $I_1 := \{x \mid f(x) \ > \ T(|x|)\}$ and $I_2 := \{x \mid f(x) \ \leq \ T(|x|)\}$.

If $f(z) > T(|z|)$ then $T^{-1}(f(z)) \ \geq \ T^{-1}(T(|z|)+1) \ \geq \ |z|$. Using the assumption that $g(n) := T(n)/n$ is monotone increasing a simple calculation yields for such x

$$g(T^{-1}(f(z))) \geq g(|z|) \qquad \Longleftrightarrow \qquad \frac{T(T^{-1}(f(z)))}{T^{-1}(f(z)))} \geq \frac{T(|z|)}{|z|}$$

$$\Longrightarrow \quad \frac{f(z)}{T^{-1}(f(z))} \geq \frac{T(|z|)}{|z|} \qquad \Longleftrightarrow \qquad \frac{f(z)}{T(|z|)} \geq \frac{T^{-1}(f(z))}{|z|} \ .$$

Then

$$\sum_{x \in I_1, \rho(x) \leq l} \frac{T^{-1}(f(x))}{|x|} \ \leq \ \sum_{x \in I_1, \rho(x) \leq l} \frac{f(z)}{T(|z|)} \ \leq \ l \ .$$

For $x \in I_2$ it holds

$$\frac{T^{-1}(f(x))}{|x|} \ \leq \ 1 \ .$$

Therefore

$$\sum_{x \in I_2, \rho(x) \leq l} \frac{T^{-1}(f(x))}{|x|} \ \leq \ l$$

□

THEOREM 4.2. *For arbitrary $\delta, \epsilon > 0$ and $T \geq (1/\delta + \epsilon)\ \mathcal{N}$ with the additional property that the function $T/\mathcal{N}$ is monotone increasing and all classes of distributions C holds*

$$\mathrm{EavTime}(T, C) \ \subseteq \ \mathrm{AvTime}(T((1+\delta)\mathcal{N}), C) \ .$$

PROOF. For $L \in \mathrm{EavTime}(T, C)$ and $\mu \in C$, let M be a DTM with $L(M) = L$ and $(\mathrm{time}_M, \mu) \in \mathrm{Eav}(T)$. Then, by a linear speedup there also exists a machine M' for L with $\mathrm{time}_{M'}(x) \ \leq \ \delta \cdot \mathrm{time}_M(x)$ for all inputs on which M spends time at least $T(|x|)$. Again, divide the inputs into the two subsets

$$\begin{aligned} I_1 \ &:= \ \{x \mid \mathrm{time}_{M'}(x) \ > \ T(|x|)\} \ , \\ I_2 \ &:= \ \{x \mid \mathrm{time}_{M'}(x) \ \leq \ T(|x|)\} \ . \end{aligned}$$

For $x \in I_1$ one can conclude as in Lemma 5

$$\frac{\mathrm{time}_{M'}(x)}{T(|x|)} \geq \frac{T^{-1}(\mathrm{time}_{M'}(x))}{|x|} \ .$$

Thus,

$$\begin{aligned} \delta \cdot l \ &\geq \ \sum_{x \in I_1 \cap X_\mu^{\leq l}} \frac{\mathrm{time}_{M'}(x)}{T(|x|)} \ \geq \ \sum_{x \in I_1 \cap X_\mu^{\leq l}} \frac{\mathrm{time}_{M'}(x)}{T(|x|)} \\ &\geq \ \sum_{x \in I_1 \cap X_\mu^{\leq l}} \frac{T^{-1}(\mathrm{time}_{M'}(x))}{|x|} \ . \end{aligned}$$

For $x \in I_2$ it holds

$$\frac{T^{-1}(\mathrm{time}_{M'}(x))}{|x|} \ \leq \ 1 \ .$$

Therefore,

$$\sum_{x \in I_1 \cap X_\mu^{\leq l}} \frac{T^{-1}(\mathrm{time}_{M'}(x))}{|x|} \ \leq \ l \qquad \text{and}$$

$$\sum_{x \in X_\mu^{\leq l}} \frac{T^{-1}(\mathrm{time}_{M'}(x))}{|x|} \ \leq \ l(1+\delta) \ .$$

□

The investigations in the next section will show that in general the second measure defines broader complexity classes. Already for the uniform ranking and a polynomial bound T it thus holds

$$\mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\}) \ \subset \ \mathrm{AvTime}(T, \{\mu_{\mathrm{uni}}\}) \ .$$

5. The Relation between Average and Worst Case Complexity Classes

From a complexity point of view sticking to a fixed distribution also does not make much sense for the following reason. If one restricts the average analysis to a simple distribution like the uniform one the best relation between average and worst case classes that seems to be obtainable is an exponential gap.

5.1. Uniform Distributions.

THEOREM 5.1.

$$\begin{aligned} \mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\}) &\subseteq \mathit{DTime}(T \cdot \mathit{EXL}) \;, \\ \mathrm{AvTime}(T, \{\mu_{\mathrm{uni}}\}) &\subseteq \mathit{DTime}(T \circ \mathit{EXL}) \;. \end{aligned}$$

PROOF. For $L \in \mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\})$ let M be a expected T-bounded DTM with $L(M) = L$. Then, for all strings z it must hold

$$\frac{\mathrm{time}_M(z)}{T(|z|)} \quad \leq \quad \sum_{x \leq z} \frac{\mathrm{time}_M(x)}{T(|x|)} \quad \leq \quad \mathrm{rank}_{\mathrm{uni}}(z) \;.$$

Thus, for every input z we get the worst case estimate

$$\mathrm{time}_M(z) \;\leq\; \mathrm{rank}_{\mathrm{uni}}(z) \cdot T(|z|) \;\leq\; \exp O(|z|) \;\cdot T(|z|) \;.$$

Now, assume $L \in \mathrm{AvTime}(T, \{\mu_{\mathrm{uni}}\})$ and let M be DTM for L such that for all strings z,

$$\frac{T^{-1}(\mathrm{time}_M(z))}{|z|} \quad \leq \quad \sum_{x \leq z} \frac{T^{-1}(\mathrm{time}_M(x))}{|x|} \quad \leq \quad \mathrm{rank}_{\mathrm{uni}}(z) \;.$$

Then, $$\mathrm{time}_M(z) \;\leq\; T(\mathrm{rank}_{\mathrm{uni}}(z) \cdot |z|) \;\leq\; T(\exp O(|z|)\,)$$ □

Note that for the EavTime-classes there is an exponential increase in the worst case bound if the time bound T is a polynomial. On the other hand, if T itself is exponential the additional factor EXL has only little influence. It is not hard to show that for small time bounds the exponential increase cannot be avoided.

THEOREM 5.2. *For any complexity bound $T \geq 2\mathcal{N}$ holds*

$$\mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\}) \quad \setminus \quad \mathit{DTime}(o(T \cdot \exp)) \quad \neq \quad \emptyset \;.$$

PROOF. Let L_1 be a unary language in

$$\mathrm{DTime}\Big(\exp(\frac{1}{2}T\cdot\exp)\Big) \ \setminus \mathrm{DTime}\Big(\exp(o(T\cdot\exp))\Big)$$

and define $L \ := \ \{xy \mid x \in L_1 \text{ and } y = 0^{2^{|x|}-|x|}\}$. Then,

$$L \ \in \ \mathrm{DTime}\left(\frac{1}{2}T\cdot\exp\right) \ \setminus \mathrm{DTime}\left(o(T\cdot\exp)\right) \ .$$

More precisely, there exists a machine M for L that rejects all strings not of the form $1^k0^{2^k-k}$ in linear time, and takes time $\frac{1}{2}T\cdot\exp$ for the remaining inputs. This yields $L \in \mathrm{EavTime}(T,\{\mu_{\mathrm{uni}}\})$ since for all l

$$\begin{aligned}
&\sum_{x\in X_\mu^{\leq l}} \frac{\mathrm{time}_M(x)}{T(|x|)} \\
\leq \quad & \sum_{x\in X_\mu^{\leq l}\setminus\{1^k0^{2^k-k}\mid k\in\mathbb{N}\}} \frac{|x|}{T(|x|)} \quad + \quad \sum_{x\in X_\mu^{\leq l}\cap\ \in\ \{1^k0^{2^k-k}\mid k\in\mathbb{N}\}} \frac{1}{2}\cdot\frac{2^{|x|}\cdot T(|x|)}{T(|x|)} \\
\leq \quad & \frac{l}{2}+\frac{1}{2}\sum_{n=1}^{\log l-1} 2^n \ \leq \ l \ .
\end{aligned}$$

Note that the previous theorem implies $L \notin \mathrm{EavTime}(o(T),\{\mu_{\mathrm{uni}}\})$. □

The AvTime-classes behave very differently in this respect. For any bound T the blowup is not negligible, it is fully exponential, as it is the case, for example, when comparing nondeterministic or probabilistic time classes with deterministic ones. This may be another indication that this concept is the more natural one for complexity investigations.

THEOREM 5.3. *For $T \geq 2\mathcal{N}$*

$$\mathrm{AvTime}\Big(T,\{\mu_{\mathrm{uni}}\}\Big) \quad \setminus \quad \mathit{DTime}\left(o(T\circ\exp\frac{\mathcal{N}}{2})\right) \quad \neq \quad \emptyset \ .$$

PROOF. Again, let L_1 be again an unary language in

$$L_1 \in \mathrm{DTime}(\exp(T\circ\exp\frac{\mathcal{N}}{2}))) \setminus \mathrm{DTime}(o(\exp(T\circ\exp\frac{\mathcal{N}}{2})))$$

and define $L := \{xy \mid x \in L_1 \text{ and } y = 0^{2^{|x|}-|x|}\}$.
$L \ \in \ \mathrm{DTime}(T\circ\exp\frac{\mathcal{N}}{2}) \setminus \mathrm{DTime}(o(T\circ\exp\frac{\mathcal{N}}{2}))$, and all strings not of the

form$1^k 0^{2^k-k}$ can be rejected in linear time by an appropriate machine M. Then, $L \in \mathrm{AvTime}(T, \{\mu_{\mathrm{uni}}\}$ follows from

$$\begin{aligned}\sum_{x \in X_\mu^{\leq l}} \frac{T^{-1}(\mathrm{time}_M(x))}{|x|} &\leq \sum_{x \in X_\mu^{\leq l} \setminus \{1^k 0^{2^k-k} \mid k \in \mathbb{N}\}} \frac{T^{-1}(|x|)}{|x|} \\ &\quad + \sum_{x \in X_\mu^{\leq l} \cap \{1^k 0^{2^k-k} \mid k \in \mathbb{N}\}} \frac{T^{-1}(T(\exp(|x|/2)))}{|x|} \\ &\leq \frac{l}{2} + \frac{1}{2} \sum_{n=1}^{\log l - 1} 2^n \leq l \ .\end{aligned}$$

Note that $L \notin \mathrm{AvTime}(T(o(\mathcal{N})), \{\mu_{\mathrm{uni}}\})$ due to Theorem 5.1. □

COROLLARY 5. *Let $T \in \omega(\mathcal{N})$ and the function $T/\mathcal{N}$ be monotone increasing, then*

$$\mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\}) \subset \mathrm{AvTime}(T(2 \cdot \mathcal{N}, \{\mu_{\mathrm{uni}}\}) \ .$$

PROOF. For $T \in \omega(\mathcal{N})$ holds: Solving the equation

$$T \circ \exp \frac{\mathcal{N}}{2} = T' \cdot \exp$$

with T' as unknown one gets a complexity bound $T' \in \omega(T)$. Construct a language L as in the last two proofs: For the AvTime classes this is done with respect to the complexity bound T, for the EavTime classes with respect to T'. In both cases we get the same language and it holds

$$\begin{aligned} L &\in \mathrm{AvTime}(T, \{\mu_{\mathrm{uni}}\}) \ , \\ L &\notin \mathrm{EavTime}(o(T'), \{\mu_{\mathrm{uni}}\}) \supseteq \mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\}) \ . \end{aligned}$$

□

If we restrict complexity classes to unary languages defined over a one letter alphabet then measuring the average complexity by the expectation obviously does not make sense because it will be exactly equal to the worst case measure. However, both of our new measures are able to provide a meaningful tool as the following result shows.

THEOREM 5.4. *There exists a unary languages L in $DTime(\mathcal{N}^2)$ whose average case complexity cannot be bounded by any function in $o(\mathcal{N}^2)$ when taking the expectation. However,*

$$L \in \mathrm{EavTime}(2\mathcal{N}, \{\mu_{\mathrm{uni}}\}) \cap \mathrm{AvTime}(2\mathcal{N}, \{\mu_{\mathrm{uni}}\}) \ .$$

PROOF. Let L_1 be a unary language in $\mathrm{DTime}(\exp \mathcal{N}^2) \setminus \mathrm{DTime}(o(\exp \mathcal{N}^2))$. Define the unary language L by

$$L \ := \ \{1^n \mid n = 2^m \text{ for some } m \in \mathbb{N} \text{ and } 1^m \in L_1\} \ .$$

Then $L \ \in \ \mathrm{DTime}(\mathcal{N}^2) \setminus \mathrm{DTime}(o(\mathcal{N}^2))$, which also implies that the time complexity measured by the expectation cannot be bounded by $o(\mathcal{N}^2)$. A simple calculation on the other hand yields

$$\begin{aligned}\sum_{x \in X_{\mu_{\mathrm{uni}}} \leq l} \frac{\mathrm{time}_M(x)}{2|x|} \ &\leq \ \sum_{x \in X_{\mu_{\mathrm{uni}}} \leq l \setminus \{x \mid |x| \in 2^k\}} \frac{|x|}{2|x|} \ + \ \sum_{x \in X_{\mu_{\mathrm{uni}}} \leq l \cap \{x \mid |x| \in 2^k\}} \frac{|x|^2}{2|x|} \\ &\leq \ \frac{l}{2} + \frac{1}{2} \sum_{n=1}^{\log l - 1} 2^n \ \leq \ l \ .\end{aligned}$$

□

This example shows that in the case of unary languages these measures tolerate larger peaks in the runtime of a machine if they do not happen too often. Thus, for unary languages we also get an averaging effect, but now over different input lengths. On the other hand, it is not hard to see that restricted to unary languages there can be at most a linear factor difference between these average and the worst case measure.

Note that this result does not contradict Theorem 1, where we have shown that for the uniform distribution the standard average time complexity based on $E_\mu(\mathrm{time}_M)$ defines the same complexity classes as $\mathrm{Eav}(T)$. In that case the average was taken with respect to a ground set of strings defined over an alphabet of size at least 2.

5.2. More Complex Distributions. Next, we will show that allowing arbitrary distributions there is no difference between the average and the worst case.

To this aim we want to construct a rank function that for any DTM M with $L(M) \notin \mathrm{DTime}(T)$ gives small ranks to inputs with long computations exceeding the time bound T. So, there is no difference between DTime and EavTime (or AvTime) with respect to this distribution. The main difficulty is to control the number of small ranks. If, for example, a machine outputs the number 1 twice this machine does not compute a rank function. On the other hand this machine must not leave any gaps in the ranking it outputs. For example, in the sequence of ranks $2, 2, 5, 5, 6, 7, 8, 9 \ldots$ rank 3 is missing. One solution to this problem is first to compute all ranks the machine outputs for smaller strings. But this requires exponential time.

Another solution is to use the following language as oracle.

Definition 12.

$$\mathbf{H_T} := \{(x, 1^i) \mid \exists z \leq x : \ time_{M_i}(z) > T(|z|)\} \ .$$

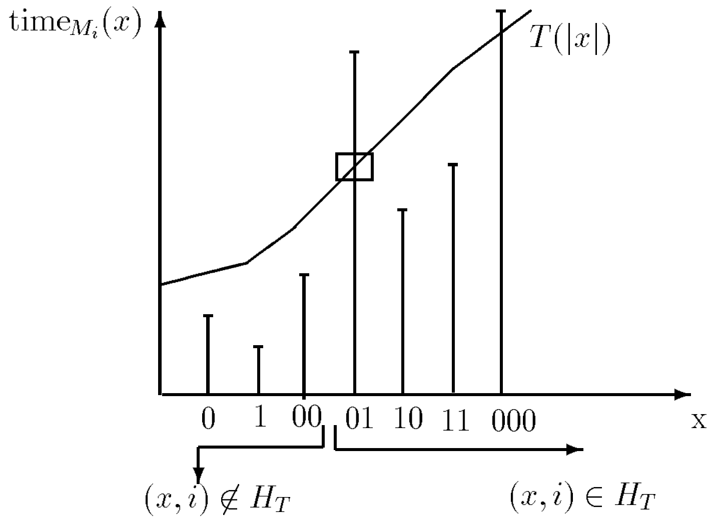

Figure 5.2: The definition of H_T.

Figure 5.2 visualize this set H_T. Note that H_T is in $\mathcal{NP}$ if T is polynomially bounded and constructible. It is also complete for $\mathcal{NP}$ since the **bounded halting problem**

$$\mathtt{NBH} \ := \ \{(x01^t0^i) \mid time_{M_i}(x) \leq t\}$$

can be reduced to it.

Let $\Omega(T^2) \ \leq \ \boldsymbol{h_T} \ \leq \ O(exp \cdot T)$ be a time bound such that $H_T \in$ DTime(h_T).

Theorem 5.5. *For all $\delta > 1$ and $T \geq 2\delta\mathcal{N}$ holds*

$$\begin{aligned} \text{EavTime}(T, h_{\delta T}\text{–rankable}) &= DTime(T), \\ \text{AvTime}(T, h_{T(\delta\mathcal{N})}\text{–rankable}) &\subseteq DTime(T(\delta\mathcal{N})). \end{aligned}$$

Proof.

1. EavTime($T, h_{\delta T}$–rankable) $\supseteq$ DTime(T)
   follows from the definition.

2. $\mathrm{EavTime}(T, h_{\delta T}\text{–rankable}) \subseteq \mathrm{DTime}(T)$:
   We will prove this inclusion indirectly. Let L be a language that is not in $\mathrm{DTime}(T)$. It suffices to show that their exists a distribution in $h_{\delta T}$–rankable with respect to which L is not average T-time bounded. To obtain the rank function of this distribution we will construct a sequence of finite sets X_i containing all inputs of finite rank and hence positive weight. Then

$$D := \bigcup_{i \in \mathbb{N}} X_i$$

   is the support of this distribution and the ranking will be the lexicographical one

$$\mathrm{rank}_D(x) \;\; := \;\; |\{z \in D \mid z \leq x\}| \quad \text{for } x \in D$$

   according to Definition 2. The X_i and the ranks of their elements will be obtained by the algorithm `long-time` defined below. We will show in Lemma 6 that a DTM can compute the rank function rank_D in time $h_{\delta T}$.

   The idea behind the construction of the X_i is as follows. X_i will be used to diagonalize against machine M_i. If $L(M_i) = L$ then the following properties will be achieved:

   (a) $\forall x \in X_i \qquad \mathrm{time}_{M_i}(x) > \delta\, T(|x|)$,
   that means each set X_i contains only inputs on which M_i spends much time (sufficiently more than the average).

   (b) $|X_i| \; > \; (\delta - 1)^{-1} \sum_{j=1}^{i-1} |X_j|$:
   these sets are chosen large enough such that there are enough long computations to prevent a good average behavior.

   (c) For all $j < l$ holds: $\qquad |x| < |y| \qquad \forall\, x \in X_j \quad \forall\, y \in X_l$,
   this way, the rank function is strictly monotone increasing and the problems with too many small ranks or holes in the ranking do not occur.

   Assume that for each i we have constructed such an X_i and that there exists a machine M_i that accepts L in average time T with respect to the distribution defined by rank_D. This would require

$$\forall l \qquad \sum_{\mathrm{rank}_D(x) \leq l} \frac{\mathrm{time}_{M_i}(x)}{T(|x|)} \;\leq\; l\;.$$

Let $l := \sum_{j=1}^{i} |X_j|$. Then using property (b), we get the contradiction

$$\sum_{\mathrm{rank}_D(x)\leq l} \frac{\mathrm{time}_{M_i}(x)}{T(|x|)} \;\geq\; \sum_{x\in X_i} \frac{\delta T(|x|)}{T(|x|)} = \delta|X_i|$$
$$= \;|X_i|(\delta-1)+|X_i| \;>\; (l-|X_i|)+|X_i| \;=\; l\;.$$

3. $\mathrm{AvTime}(T, h_{T(\delta\cdot\mathcal{N})}\text{–rankable}) \;\subseteq\; \mathrm{DTime}(T(\delta\mathcal{N}))$:

   As above, consider any language $L \notin \mathrm{DTime}(T(\delta\mathcal{N}))$. Now replace the first property of X_i by

   (a') $\forall x\in X_i \quad \mathrm{time}_{M_i} > T(\delta|x|)$.

   Assume that there exists a Turing machine M_i with $L(M_i) = L$ and $(\mathrm{time}_{M_i}, \{\mathrm{rank}_D\}) \in \mathrm{Av}(T)$, that means

$$\forall l \qquad \sum_{\mathrm{rank}_D(x)\leq l} \frac{T^{-1}(\mathrm{time}_{M_j}(x))}{|x|} \;\leq\; l$$

   Again, for $l = \sum_{j=1}^{i} |X_j|$ we derive a contradiction

$$\sum_{\mathrm{rank}_D(x)\leq l} \frac{T^{-1}(\mathrm{time}_{M_i}(x))}{|x|} \;\geq\; \sum_{x\in X_i} \frac{T^{-1}(T(\delta|x|))}{|x|} \;=\; \delta|X_i| \;>\; l\;.$$

The sequence X_i is constructed as follows. Define the iterated exponential function $\mathrm{itexp}(n) \;:=\; \underbrace{\exp((\cdots\exp(}_{n \text{ times}}1)\cdots))$, its inverse itlog $:=$ itexp^{-1} and strings $z_i := 1^{\mathrm{itexp}(i)}$. Note that for any string x with respect to the lexicographical ordering $z_{\mathrm{itlog}(|x|)-1} \;<\; x \;\leq\; z_{\mathrm{itlog}(|x|)}$.

Let M^* be a DTM for L. To make the computation of the ranks efficient for each machine M_i we we will use another machine with identical time behavior for large inputs, but a linear time bound for small inputs. It is irrelevant whether this machine accepts the same language as M_i. More formally, let $c : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ be a function such that for all i

$$\mathrm{time}_{M_{c(i,k)}}(x) := \begin{cases} \mathrm{time}_{M_i}(x) & \text{if } x \geq z_k,\\ |x| & \text{if } x < z_k.\end{cases}$$

It is easy to see that such indices $c(i,k)$ of size bounded by $O(i\cdot k)$ can be computed time $O(i\cdot k)$.

Below we describe a recursive algorithm `long-time`, which on input x outputs a tupel $(\texttt{index}, s, j, r)$. The meaning of these components are as follows. `index` denotes the index of the set X_{index} into which x might be inserted. j counts the number of strings lexicographically smaller than or equal to x that are contained in X_{index}, while s counts those in D. r equals $\text{rank}_D(x)$. Thus in particular, the last component gives the rank function. The other information will be needed in the recursive calls to achieve the properties discussed above.

As a subroutine we will check whether a machine M_i behaves different from M^* for any inputs z much smaller than the current input x. This is done by the following procedure.

$\texttt{error}(i, x)$
<u>for all</u> z <u>with</u> $|z| \leq \log\log|x|$
  simulate machines M^* and M_i on input z for $|x|/(\log|x|)^2$ steps;
  <u>if</u> either (both accept) or (both reject or have not terminated)
    <u>then</u> <u>return</u> `true`
<u>return</u> `false`

The main algorithm looks as follows.

$\texttt{long-time}(x)$
  <u>if</u> $|x| = 1$ <u>then</u> <u>return</u> $(0, 0, 0, \infty)$;
  $(\texttt{index}, j, s, r) \ := \ \texttt{long-time}(z_{\text{itlog}(|x|)-1})$
  case 1: $\text{error}(\texttt{index}, x) = \texttt{true}$ <u>then</u>
    <u>if</u> $(c(\texttt{index}, \text{itlog}(|x|-1)), x-1) \in H_{\delta T}$
      <u>then</u> <u>return</u> $(\texttt{index}+1, 0, s+1, \infty)$;
      <u>else</u> <u>return</u> $(\texttt{index}+1, 0, s, \infty)$;
  case 2: $(c(\texttt{index}, \text{itlog}(|x|-1)), x-1) \in H_{\delta T}$ <u>then</u>
    <u>return</u> $(\texttt{index}, j+1, s+1, \infty)$;
  case 3: $\text{time}_{M_{\text{index}}}(x) \ \leq \ \delta T(|x|)$ <u>then</u>
    <u>return</u> $(\texttt{index}, j, s, \infty)$;
  case 4: $j > \frac{s-j}{\delta-1}$ <u>then</u>
    include x into X_{index} and <u>return</u> $(\texttt{index}+1, 0, s+1, s+1)$;
  case 5: <u>else</u>
    include x into X_{index} and <u>return</u> $(\texttt{index}, j+1, s+1, s+1)$.

In case 1 the index can be increased since $L(M_i) \neq L$ and one does not have to worry about machine M_i anymore. If case 2 applies there is already one string from the interval $]z_{\mathrm{itlog}(|x|)-1}, z_{\mathrm{itlog}(|x|)}]$ included into X_{index} and we will not choose anymore. In the last two cases M_i spends too much time on input x and hence this input will be put into X_{index}. If we now have enough such strings (case 4) the index will be increased, otherwise we still have to find more strings for this set.

This procedure will not get stuck at some index i because this would imply that only for finitely many x holds $\mathrm{time}_{M_i}(x) > \delta T(|x|)$. But then another machine would accept L in worst case time $\delta \cdot T$. By a linear speedup L could also be accepted in worst case time T, a contradiction to the choice of L.

By construction, the sets X_i fulfill the desired conditions (a), (b) and (c).

LEMMA 6. *$\mathrm{rank}_D \in O(h_{\delta T})$–rankable*

PROOF. Let $d(n)$ be an upper bound on the run time of $\texttt{long} - \texttt{time}$ on input of length n.

1. The computation of $\texttt{long-time}(z_{\mathrm{itlog}(|x|)-1})$ requires at most $d(\log |x|)$ steps.

2. $\texttt{error}(\texttt{index}, x)$ can be computed in linear time, since $\texttt{index} \leq \mathrm{itlog}|x|$.

3. The simulation of M_{index} for $\delta T(|x|)$ steps costs time $\delta \cdot T(|x|) \cdot \mathrm{itlog}(|x|)^2 \leq h_{\delta T}(|x|)$.

4. To simulate the oracle, time $h_{\delta T}(|x|)$ is sufficient by definition of $h_{\delta T}$.

Therefore, $\mathrm{rank}_D \in O(h_{\delta T})$–rankable. □

The proof for the Av-measure is almost identical replacing δT by $T(\delta \cdot \mathcal{N})$ in the construction above. □

Miltersen has shown that there exists a distribution μ malign for $\mathrm{DTime}(\mathcal{N}^k)$ with respect to the expected time measure, which can be computed in polynomial time with an $\Sigma_2^{\mathcal{P}}$–oracle (Miltersen 93). The theorem above shows that an $\mathcal{NP}$–oracle suffices.

COROLLARY 6.

$$\mathrm{EavTime}(T, C) \;=\; \mathit{DTime}(T)$$

for a set C of distributions that are $O(T \cdot \log^2 T)$–rankable with an oracle for $H_{T(\delta\mathcal{N})}$ ($\delta > 1$).

Using the fact that the time bound h_T grows at most like $T(n) \cdot 2^n$ and that for polynomial bounds the constant δ can be omitted one obtains

COROLLARY 7. *For complexity bounds T, T' with $T' \in POL$ holds:*

$$\begin{aligned}\text{EavTime}(T, (T \cdot EXL)\text{–}rankable) &= DTime(T),\\ \text{AvTime}(T', (T' \cdot EXL)\text{–}rankable) &= DTime(T').\end{aligned}$$

Notice that for the stricter average measure the equality is only claimed for polynomial bounds T'. The motivation for considering average measures other than the expectation was to keep the concept of polynomial time reductions. This aim seems to imply that the precision of the measure has to be weekened. Levin's approach is very coarse grain, while the one presented here will turn out to be as precise as the worst case measure in the most interesting range of polynomial bounds. But for time bounds much larger than polynomials the situation becomes somewhat different.

THEOREM 5.6. *For $T \geq EEXL$ and the set of all distributions U holds:*

$$DTime(T) \subset \text{AvTime}(T, U)\ .$$

PROOF.
The relation "$\subseteq$" follows from the definition.
That the deterministic class is strictly contained in the average time class can be seen as follows. Let $T = \exp\exp$ be the double exponential time bound and $L_E := \{1^{2^i} \mid i \in \mathbb{N}\}$.
Let $L \subseteq L_E$ be a language in $\text{DTime}(T^2) \setminus \text{DTime}(T)$ and μ be an arbitrary distribution. Let x, y be the elements in L_E with the two smallest ranks according to μ. One can construct a machine M for L that takes linear time for all inputs in $\overline{L_E} \cup \{x, y\}$ and time T^2 else. This means for input x that

$$\frac{T^{-1}(\text{time}_M(x))}{|x|} \quad\leq\quad \frac{\log\log|x|}{|x|} \quad\leq\quad 1/2$$

and similarly for y.

Then, for each l let $e_l := |X_\mu^{\leq l} \cap L_E|$:

$$\begin{aligned}\sum_{z \in X_\mu^{\leq l}} \frac{T^{-1}(\text{time}_M(z))}{|z|} &\leq 1 + \sum_{z \in X_\mu^{\leq l} \setminus L_E} \frac{T^{-1}(|z|)}{|z|} + \sum_{1^{2^j} \in X_\mu^{\leq l} \setminus \{x,y\}} \frac{T^{-1}(\text{time}_M(1^{2^j}))}{|1^{2^j}|}\\ &\leq 1 + \sum_{z \in X_\mu^{\leq l} \setminus L_E} 1 + \sum_{1^{2^j} \in X_\mu^{\leq l} \setminus \{x,y\}} \frac{\log\log(2^{2 \cdot 2^{2^j}})}{2^j}\\ &= 1 + l - e_l + \sum_{1^{2^j} \in X_\mu^{\leq l} \setminus \{x,y\}} 1 + \frac{1}{2^j}\\ &\leq l - e_l + 1 + (e_l - 2) + 1 \quad = \quad l\ .\end{aligned}$$

Thus, $L \in \mathrm{AvTime}(T, \{\mu\})$ for all distributions μ. $\square$

6. Time Hierarchies of Average Classes

For average complexity classes with a fixed bound on the rankability of the distribution we get tight hierarchy results. First we show that even under uniform distribution we cannot always solve a problem of a higher worst case complexity class. This substantially improves theorem 4.2 in Miltersen 93, which gives a weaker separation for the expected measure.

THEOREM 6.1. *For time-constructible and monotone increasing time bounds T_1, T_2 with $T_1 \leq o(T_2)$ holds*

$$\begin{aligned} \mathit{DTime}(T_2) \ \setminus \ \mathrm{EavTime}(T_1, \{\mu_{\mathrm{uni}}\}) &\neq \emptyset \ , \\ \mathit{DTime}(T_2) \ \setminus \ \mathrm{AvTime}(T_1, \{\mu_{\mathrm{uni}}\}) &\neq \emptyset \ . \end{aligned}$$

PROOF. Define a function $f : \mathbb{N} \to \mathbb{N}$ as follows.

$$\begin{aligned} f(0) &:= 1 \\ n_i &:= \min\left\{ b \geq f(i) \,\middle|\, \forall j \in [b; 2b+1] \quad \frac{T_2(j)}{T_1(j)} \geq i \right\} \\ f(i+1) &:= 2^{2n_i+1} \end{aligned}$$

With the help of f we define a diagonal language L by

$$\begin{aligned} \boldsymbol{L} \ := \ \{x \mid & \text{ for } i \text{ with } f(i) \leq |x| < f(i+1) \text{ holds:} \\ & i \cdot T_1(|x|) \leq T_2(|x|) \quad \text{and} \\ & \left[\ \mathrm{time}_{M_i}(x) > 2 \cdot T_1(|x|) \quad \text{or} \quad x \notin L(M_i)\ \right] \}. \end{aligned}$$

The motivation for these definitions is as follows:

- Like in a normal diagonalization a string x belongs to L if a corresponding machine M_i on input x computes longer than $2 \cdot T_1(|x|)$ steps or rejects.
- To guarantee $L \in \mathrm{DTime}(T_2)$ such a simulation of M_i should only be performed if there is enough time, that means $i \cdot T_1(|x|) \leq T_2(|x|)$.
- To exceed the average time bound T_1 we let the index i grow very slowly. So, we get for each index i at least as many finite ranks (relevant for the diagonalization) as for all indices $1 \ldots i-1$ together. This is achieved by the rapid growth of the n_i.

- To be able to compute the inverse of the resulting function f in time $T_2(|x|)$ we choose $f(i+1)$ as 2^{2n_i+1}.

1. $L \in \mathrm{DTime}(T_2)$:

   To decide L by a DTM M, first for an input x compute the corresponding i such that $f(i) \leq |x| < f(i+1)$. For this aim the machine successively computes all $f(j)$ for $j \leq |x|$. This is easy in the given time bound since f grows very fast ($f \in \omega(\mathrm{itexp})$). For a given $f(j)$, $f(j+1)$ is computed by incrementing two counters b, j until b fulfills the condition for n_j. If $j > \log|x|$, the algorithm halts: the last computed value of f is $f(i)$. This costs at most $(\log f(j+1) - f(j)) \cdot T_2(\log f(j+1)) \leq T_2(j+1)$ many steps for almost all j. It is not necessary to compute $f(i+1)$. Hence, i is computed in time $O(T_2)$.

   If $i \cdot T_1(|x|) > T_2(|x|)$ the machine rejects x.

   Otherwise, M simulates $2 \cdot T_1(|x|)$ steps of the machine M_i on input x. By assumption this can be done in time $i \cdot T_1(|x|) \leq T_2(|x|)$. If M_i accepts x within that many steps M rejects x, else it accepts.

2. $L \notin \mathrm{EavTime}(T_1, \{\mu_{\mathrm{uni}}\})$:

   Assume that $L \in \mathrm{EavTime}(T_1, \{\mu_{\mathrm{uni}}\})$. Let M_i be a machine that accepts L. By construction of the sequence n_i we have that for all $x \in [n_i, 2n_i+1]$: $i \cdot T_1(|x|) \leq T_2(|x|)$. Hence, the first condition in the definition of L is always fulfilled for such x. Now $L = L(M_i)$ implies that for such x the case $x \notin L(M_i)$ in the definition cannot occur, hence $\mathrm{time}_{M_i}(x) > 2 \cdot T_1(|x|)$. We claim that the average time complexity of M_i with respect to the uniform distribution is larger than T_1 both for the eAv and the Av–measure. Otherwise, for the first measure we would get the following contradiction. Let $l = 2^{2n_i+1} - 1$, then

$$\begin{array}{rcll}
\displaystyle\sum_{x \in X^{\leq l}_{\mu_{\mathrm{uni}}}} \frac{\mathrm{time}_{M_i}(x)}{T_1(|x|)} & \leq & l & \Rightarrow \\
\displaystyle\sum_{2^{n_i} \leq \mathrm{rank}_{\mathrm{uni}}(x) < 2^{2n_i+1}} \frac{\mathrm{time}_{M_i}(x)}{T_1(|x|)} & \leq & l & \Rightarrow \\
\displaystyle\sum_{2^{n_i} \leq \mathrm{rank}_{\mathrm{uni}}(x) < 2^{2n_i+1}} 2 & \leq & 2^{2n_i+1} & \Rightarrow \\
2 \cdot (2^{2n_i+1} - 2^{n_i}) & \leq & 2^{2n_i+1} & \Rightarrow \\
2^{2n_i+1} & \leq & 2 \cdot 2^{n_i} &
\end{array}$$

3. $L \notin \mathrm{AvTime}(T_1, \{\mu_{\mathrm{uni}}\})$:

For the other measure a contradiction is derived similarly.

$$\begin{aligned}
\sum_{x \in X^{\leq l}_{\mu_{\mathrm{uni}}}} \frac{T_1^{-1}(\mathrm{time}_{M_i}(x))}{|x|} &\leq l && \Rightarrow \\
\sum_{2^{n_i} \leq \mathrm{rank}_{\mathrm{uni}}(x) < 2^{n_i+1}} \frac{T_1^{-1}(\mathrm{time}_{M_i}(x))}{|x|} &\leq l && \Rightarrow \\
\sum_{2^{n_i} \leq \mathrm{rank}_{\mathrm{uni}}(x) < 2^{2n_i+1}} \frac{|x|+1}{|x|} &\leq 2^{2n_i+1} && \Rightarrow \\
\sum_{2^{n_i} \leq \mathrm{rank}_{\mathrm{uni}}(x) < 2^{2n_i+1}} 1 + \frac{1}{|x|} &\leq 2^{2n_i+1} && \Rightarrow \\
(2^{2n_i+1} - 2^{n_i}) \cdot \left(1 + \frac{1}{2^{n_i}}\right) &\leq 2^{2n_i+1} && \Rightarrow \\
2 \cdot 2^{n_i} &\leq 2^{n_i} + 1 && \square
\end{aligned}$$

Now observe that if $V_1 \leq V_2$ then V_1–rankable $\subseteq V_2$–rankable and hence

$$\mathrm{EavTime}(T, V_1\text{–rankable}) \supseteq \mathrm{EavTime}(T, V_2\text{–rankable}) \ .$$

Therefore, for $V_1 \geq \mathcal{N}$

$$\begin{aligned}
\mathrm{DTime}(T) &\subseteq \mathrm{EavTime}(T, V_1\text{–rankable}) \\
&\subseteq \mathrm{EavTime}(T, \{\mu_{\mathrm{uni}}\})
\end{aligned}$$

and similarly for the other measure. This implies

COROLLARY 8. *Let $T_1, V \geq (1+\epsilon)\mathcal{N}$ and $T_2 \in \omega(T_1)$. Then*

$$\begin{aligned}
\mathrm{EavTime}(T_1, V\text{–}\mathit{rankable}) &\subset \mathrm{EavTime}(T_2, V\text{–}\mathit{rankable}) \ , \\
\mathrm{AvTime}(T_1, V\text{–}\mathit{rankable}) &\subset \mathrm{AvTime}(T_2, V\text{–}\mathit{rankable}) \ .
\end{aligned}$$

7. Distribution Hierarchies of Average Classes

In the previous section we have seen that like in the worst case any slight increase in the average time bound gives more computational power. Now we want to investigate the same question with respect to the complexity of the distributions. Will a more severe restriction on the set of distributions, for which one requires a good average behavior, allow to solve more algorithmic problems within a given average time bound?

7.1. Diagonalization with respect to Sets of Rank Functions. Again a proof for this property is likely to use a special kind of diagonalization, but there is a slight technical problem arises. In general, it is not possible to enumerate exactly those machines that compute rank functions of a certain time complexity because this task is equivalent to the halting problem. We circumvent this difficulty by considering a broader class of rank functions.

DEFINITION 13. *A* **pseudo rank function** *is function* $r : \Sigma^* \rightarrow \mathbb{N} \cup \{\infty\}$ *with the property that for all* $l \in \mathbb{N}$ *holds:* $|\{x|r(x) \leq l\}| \leq l$.

An example of a pseudo rank function that is not a rank function is the function $1 \mapsto 4,\ 2 \mapsto 5,\ 3 \mapsto 6, \ldots$. Some rank values never appear, but it does not happen that too many small ranks are generated. One cannot enumerate all machines computing pseudo rank functions either, but at least one can construct a dynamic set of candidates such that each machine fulfilling this property will eventually be contained in this set, and each machine not fulfilling this property will eventually be thrown out.

DEFINITION 14. *Let V be a time bound. A* **V-rank-accumulator** *is a DTM M that on input 1^n computes a set of machine indices $I_n \subset \mathbb{N}$ with the following property:*

- $\forall\, i \in \mathbb{N}$ *holds:* M_i *computes a pseudo rank function and is V-time bounded* $\iff$ $\exists\, n_0 \quad \forall n \geq n_0 \quad i \in I_n$.

LEMMA 7. *For every complexity bound V there exists a V-rank-accumulator M. Furthermore, M can be made to run in linear time.*

PROOF. Let $c : \mathbb{N} \rightarrow \mathbb{N}$ be a monotone, unbounded function that can be computed in linear time. The following algorithm defines a V-rank-accumulator.

```
on input 1^n compute c(n) and define I := [1 ... c(n)]
for all i ∈ I
    R_i := ∅
    for all x ∈ Σ^{≤c(n)}
        simulate machine M_i on input x for at most V(|x|) steps
        if M_i has not terminated by that time
            then remove i from I
            else add its output to the multiset R_i
    for all l ∈ R_i
        if |{r ∈ R_i | r ≤ l}| > l then remove i from I
    return I
```

The correctness of this algorithm can be seen easily. Its time complexity on input 1^n is given by

$$T_{V,c}(n) \;\leq\; O\Big(c(n)\cdot|\Sigma|^{c(n)+1}\cdot V(c(n))\Big)$$

(remember that complexity bounds like V were assumed to be monotone increasing). For any V one can find a monotone and unbounded function c such that $T_{V,c}$ grows at most linearly, and $c(n)$ can be computed in time $O(n)$. For example, the following function has these properties

$$c(n) \;:=\; \min\left\{V^{-1}(\sqrt{n}), \frac{\log n}{3\log|\Sigma|}\right\} \;. \qquad \square$$

Lemma 8. *For all complexity bounds V_1, V_2 with $\mathcal{N} \leq V_1 \leq o(V_2)$ there exists an infinite language L with the following properties:*

- *$rank_L \in V_2$–rankable,*
- *for all $R \in V_1$–rankable : $\quad rank_L(x) < \infty \implies R(x) \geq rank_L(x)^2$ for almost all x.*

Proof. The language L to be constructed will be a subset of 1^*. Remember that according to our general assumption on complexity bounds V_2 is monotone and time-constructible. We start with a sequence $n_1 = 1 < n_2 < n_3 < \ldots \subseteq \mathbb{N}$ with the following properties:

- $V_2(n_{l+1}) \geq 2\cdot V_2(n_l)$
- for any $m \in \mathbb{N}$ the element n_l with $n_l \leq m < n_{l+1}$ can be computed in time $O(V_2(m))$.

For example, such a sequence can be obtained as follows: Assume n_l has been defined. Then, for $k = 1, 2, 3, \ldots$ consider the sequence $V_2(2^k\cdot n_l)$ till the first k such that

$$V_2(2^k\cdot n_l) \;\geq\; 2^k\cdot V_2(n_l)\;.$$

It is easy to see that such a value, let us denote it by $\bar{k}$, must exist. Otherwise, the partial sums of V_2 were bounded quadratically, which contradicts the condition that V_2 grows faster than linear. Considering n_l fixed and m growing, one would get

$$\begin{aligned}\sum_{a=1}^{n_l+2^m} V_2(a) \quad &= \quad \sum_{a=1}^{n_l-1} V_2(a) \;+\; \sum_{a=n_l}^{n_l+2^m} V_2(a)\\ &\leq \quad O(1) \;+\; \sum_{a'=1}^{m} 2^{a'-1}\cdot V_2(2^{a'}\cdot n_l)\end{aligned}$$

$$\leq \quad O(1) \; + \; \sum_{a'=1}^{m} 2^{a'-1} \cdot 2^{a'} \cdot V_2(n_l)$$
$$\leq \quad O(1) \; + \; 2^{2m} \cdot V_2(n_l) \;\leq\; O((n_l + 2^m)^2) \; .$$

Define $n_{l+1} := 2^{\bar{k}} \cdot n_l$. Then this sequence obviously fulfills the first condition. Since V_2 is time constructible, given n_l the next value n_{l+1} can be computed in time

$$\sum_{k=1}^{\bar{k}} V_2(2^k \cdot n_l) \;\leq\; \sum_{k=1}^{\bar{k}-1} 2^k \cdot V_2(n_l) + V_2(2^{\bar{k}} \cdot n_l) \;\leq\; 2 \cdot V_2(n_{l+1}) \; .$$

Thus, the whole sequence $n_1, \ldots, n_{l+1}$ can be computed in time $O(V_2(n_{l+1}))$, and for $m = n_{l+1}$ the second condition holds, too. For arbitrary m first compute $V_2(m)$ and then set a time limit of $O(V_2(m))$. Start computing the sequence $n_1, \ldots$ until either n_{l+1} has been found or the time limit is over. In the second case, the last value computed is the desired n_l. Define

$$x_l := 1^{n_l} \; .$$

The following algorithm computes rank_L, and thus implicitly defines L. On input x_l a triple (j, k, r) is generated where r denotes the rank of x_l in L. Thus $x_l \in L$ iff $r < \infty$. j and k are additional parameters that are needed in this recursive procedure.

$\texttt{RANKL}(x)$
<u>for</u> input $x = 1^m$ find l such that $m = n_l$;
<u>if</u> l does not exist <u>then</u> <u>return</u> $(-, -, \infty)$;
<u>if</u> $l = 1$
    <u>then</u> <u>return</u> $(1, 1, 1)$
    <u>else</u> $(j, k, r) \;:=\; \texttt{RANKL}(x_{l-1})$;
let I_k be the output of a V_1-rank accumulator on input 1^k;
$I := I_k \cap [1..b]$, where $b := \min\left\{j, \frac{V_2(m)}{6 \cdot V_1(m)}\right\}$;
<u>for all</u> $i \in I$
    simulate machine M_i on input x for at most $V_1(|x|)$ steps;
    <u>if</u> by that time M_i has output a rank less than $(r+2)^2$
        <u>then</u> <u>return</u> $(j, k+1, \infty)$;
<u>return</u> $(j+1, j, r+1)$.

We have to show that the language L constructed this way possesses the properties stated above.

1. $\mathrm{rank}_L \in V_2$–rankable:
   Assume inductively that on input x_{l-1} the time complexity of `RANKL` is bounded by $V_2(n_{l-1})$. Now consider $x = 1^m$ with $m > n_{l-1}$. We may assume that determining n_{l-1}, resp. n_l takes time at most $\frac{1}{6}V_2(m)$.

   If $m = n_l$ then due to the properties of this sequence the time for the recursive call of $\texttt{RANKL}(x_{l-1})$ is bounded by $\frac{1}{2}V_2(n_l)$. Also, we may assume that the output of the rank accumulator can be obtained in time $\frac{1}{6}V_2(n_l)$. The time needed for the simulation of all machines in I is bounded by

$$b \cdot V_1(m) \;\leq\; \frac{V_2(m)}{6 \cdot V_1(m)} \cdot V_1(m) \;=\; \frac{1}{6}V_2(n_l) \;.$$

   This proves that rank_L can be computed in time V_2.

2. L is infinite:
   Assume that for some input x_l a finite rank r is computed by `RANKL`. As long as the next candidates $x = x_{l+1}, x_{l+2}, \ldots$ do not get a finite rank the variable j stays fixed and k is increased each time. By definition of a V_1-rank-accumulator, eventually for some k the set $I = I_k \cap [1..b]$ with $b = \min\{j, V_2(|x|)/6V_1(|x|)\}$ only contains indices of machines that compute a pseudo rank function. Any one of them can hurt the condition "output"$< (r+1)^2$ only a finite number of times. Thus, eventually the rank $r+1$ will be assigned to some successor of x_l.

3. For all $R \in V_1$–rankable and for almost all x:

$$\mathrm{rank}_L(x) < \infty \qquad \Longrightarrow \qquad R(x) \geq (\mathrm{rank}_L(x) + 1)^2 \;.$$

   The index of a V_1-time bounded machine M computing R will eventually be considered in the *for* -loop on some input x_l. Then for all inputs x_s with $s \geq l$ holds:
   if `RANKL` generates a finite rank r it will not be larger than $\sqrt{R(x_s)} - 1$.
   □

For a language L with rank function $\mathrm{rank}_L(x)$ define

$$R_L(n) \;:=\; \max\{\mathrm{rank}_L(x) \;\big|\; |x| \leq n,\ \mathrm{rank}_L(x) < \infty\} \;.$$

LEMMA 9. *Let T_1, V_1, V_2 be complexity bounds with the properties $\mathcal{N} \leq V_1 \leq o(V_2)$ and $V_2 \leq O(T_1)$. Then there exists a language $L \in DTime(V_2)$ and a complexity bound $T_2 \in \omega(T_1)$ such that*

- *$\mathrm{rank}_L \in V_2$–rankable $\setminus V_1$–rankable and*
- *$P \cap L \in \mathrm{EavTime}(T_1, V_1$–rankable$)$ for all $P \in \mathrm{DTime}(T_2)$.*

PROOF. Let L be a language with the properties as described in lemma 8, and define $T_2(n) := T_1(n) \cdot R_L(n)$.

1. $\mathrm{rank}_L \in V_2$–rankable$\setminus V_1$–rankable follows immediately from the construction of L since any rank function of complexity at most V_1 will infinitely often generate rank values larger than rank_L.

2. $\forall P \in \mathrm{DTime}(T_2) \quad P \cap L \in \mathrm{EavTime}(T_1, V_1$–rankable$)$:

   Let r be the rank function of a distribution that is V_1–rankable, and choose $P \in \mathrm{DTime}(T_2)$. To decide whether $x \in P \cap L$ do the following:

   - For the finitely many inputs x with $r(x) \le (\mathrm{rank}_L(x)+1)^2$ prepare a decision table beforehand to get the result in linear time.
   - Otherwise, test whether $x \in L$. Reject, if not.
     Since $L \in \mathrm{DTime}(V_2)$ and, by assumption, $V_2 \le O(T_1)$ this question can be decided in time $T_1(|x|)$ (using a linear speedup if necessary).
   - In the remaining case it holds $x \in L$ and thus $\mathrm{rank}_L(x) < \infty$, and furthermore $r(x) > (\mathrm{rank}_L(x)+1)^2$.
     Then, use a T_2–time bounded algorithm to decide whether $x \in P$.

   To estimate the average time complexity of a machine M performing this computation we get for any $l \in \mathbb{N}$:

$$
\begin{aligned}
\sum_{r(x)\le l} \frac{\mathrm{time}_M(x)}{2\cdot T_1(|x|)} &\le \sum_{\substack{r(x)\le l \text{ and} \\ r(x)\le(\mathrm{rank}_L(x)+1)^2}} \frac{T_1(|x|)}{2\cdot T_1(|x|)} \\
&\quad + \sum_{\substack{r(x)\le l \text{ and} \\ r(x)>(\mathrm{rank}_L(x)+1)^2}} \frac{T_2(|x|)}{2\cdot T_1(|x|)} \\
&\le \frac{l}{2} + \sum_{\mathrm{rank}_L(x)<\sqrt{l}-1} \frac{T_1(|x|)\cdot R_L(|x|)}{2\cdot T_1(|x|)} \\
&\le \frac{l}{2} + \frac{1}{2}\cdot \sum_{i=1}^{\lfloor\sqrt{l}\rfloor-1} i \le l\,.
\end{aligned}
$$

Therefore $(\mathrm{time}_M, r) \in \mathrm{Eav}(2 \cdot T_1)$. By a linear speedup, one can find a machine M' with $L(M') = L(M) = P \cap L$ and $(\mathrm{time}_{M'}, r) \in \mathrm{Eav}(T_1)$. $\square$

LEMMA 10. *Let $\delta > 1$ and T_1, V_1, V_2 be complexity bounds with $\mathcal{N} \leq V_1 \leq o(V_2)$ and $V_2(\delta\mathcal{N}) \leq O(T_1)$. Then, there exists a language $L \in DTime(V_2)$ and a complexity bound $T_2 \in \omega(T_1)$ such that*

- *$\mathrm{rank}_L \in V_2$–rankable $\setminus$ V_1–rankable and*
- *$P \cap L \in \mathrm{AvTime}(T_1, V_1$–rankable$)$ for all $P \in DTime(T_2)$.*

PROOF. Use the same L as in the previous lemma. The time bound T_2 is now defined as:

$$T_2(n) \ := \ T_1\left(\left\lfloor n \cdot R_L(n) \cdot \left(1 - \delta^{-1}\right)\right\rfloor\right) \ .$$

We already know that $\mathrm{rank}_L \in V_2$–rankable $\setminus$ V_1–rankable. It remains to show that for all $P \in \mathrm{DTime}(T_2)$:

$$P \cap L \ \in \ \mathrm{AvTime}(T_1(2 \cdot \mathcal{N}), V_1\text{–rankable}) \ .$$

For a given $P \in \mathrm{DTime}(T_2)$ and for a given rank function $r \in V_1$–rankable construct a machine M as above.

- For the finitely many x with $r(x) \leq (\mathrm{rank}_L(x)+1)^2$ the answer is encoded into the machine and obtained in linear time.
- M tests if $x \in L$ and rejects if not.
  This now takes time at most $O(T_1(|x|/\delta))$.
- In the remaining case, $\mathrm{rank}_L(x) < \infty$ and $r(x) > (\mathrm{rank}_L(x) + 1)^2$, decide $x \in P$ within at most $T_2(|x|)$ additional steps.

Then for all $l \in \mathbb{N}$:

$$\begin{aligned}
\sum_{r(x)\leq l} \frac{{T_1}^{-1}(\mathrm{time}_M(x))}{|x|} \quad &\leq \quad \sum_{\substack{r(x)\leq l \text{ and} \\ r(x)\leq(\mathrm{rank}_L(x)+1)^2}} \frac{{T_1}^{-1}(T_1(|x|/\delta))}{|x|} \\
&\quad + \sum_{\substack{r(x)\leq l \text{ and} \\ r(x)>(\mathrm{rank}_L(x)+1)^2}} \frac{{T_1}^{-1}(T_2(|x|))}{|x|} \\
&\leq \quad \frac{l}{\delta} + \sum_{\mathrm{rank}_L(x)\leq\sqrt{l}-1} \frac{R_L(|x|) \cdot |x| \cdot (1 - \frac{1}{\delta})}{|x|} \\
&\leq \quad \frac{l}{\delta} + \left(1 - \frac{1}{\delta}\right) \cdot \sum_{i=1}^{\lfloor\sqrt{l}\rfloor-1} i \ \leq \ l \ .
\end{aligned}$$

Therefore $(\mathrm{time}_M, r) \in \mathrm{Av}(T_1)$. □

7.2. The Separation.

LEMMA 11. *Let $\mathcal{N} \leq V_2 \leq o(T_1)$ and $T_1 \leq o(T_2)$. Then for all $\delta > 1$ and for all infinite $L \in DTime(V_2)$, there exists a $P \in DTime(T_2)$ such that*

$$P \cap L \notin \mathrm{EavTime}(T_1, \{rank_L\}) .$$

PROOF. Define

$$\begin{aligned} f(0) &:= 1 , \\ n_i &:= \min\left\{ b \geq f(i) \;\middle|\; \forall\, j \in [b \ldots R_L^{-1}(R_L(b) + 2^{2b+1})] : \frac{T_2(j)}{T_1(j)} \geq i \right\}, \\ f(i+1) &:= 2^{2n_i+1} , \end{aligned}$$

and a diagonal language like in the proof of theorem 6.1 by

$$\begin{aligned} P := \{x \mid & \text{ for } i \text{ with } f(i) \leq |x| < f(i+1) \text{ holds:} \\ & i \cdot T_1(|x|) \leq T_2(|x|) \quad \text{and} \\ & [\mathrm{time}_{M_i}(x) > 2 \cdot T_1(|x|) \quad \text{or} \quad x \notin L(M_i)]\ \} \ . \end{aligned}$$

Since $T_1 \leq o(T_2)$ the sequence n_i is well defined (always $< \infty$).

1. $P \in \mathrm{DTime}(T_2)$:

   On input x, first compute all $f(j)$ and n_j smaller than $\log|x|$. Because of the strong growth of f and the time constructibility of all time bounds this can easily be done in time $T_2(|x|)$.

   Compute i with $f(i) \leq |x| < f(i+1)$ in time $O(|x|)$ using the last computed n_i.

   Test whether $i \cdot T_1(|x|) \leq T_2(|x|)$. If this does not hold reject, otherwise simulate the machine M_i for $2 \cdot T_1(|x|)$ steps. Accept iff M_i does not accept within that period.

   This simulation costs at most $2 \cdot T_1(|x|) \cdot i \leq 2 \cdot T_2(|x|)$ many steps.

2. $P \cap L \notin \mathrm{EavTime}(T_1, \{\mathrm{rank}_L\})$:

   Assume that M_i computes $P \cap L$ in expected average time T_1 with respect to the ranking rank_L. By definition of n_i and R_L it holds for all i

   $$|\{x \in L |\ f(i) \leq |x| < f(i+1) \text{ and } T_1(|x|) \cdot i \leq T_2(|x|)\}| \geq R_L(n_i) + 2^{2n_i+1}$$

Furthermore, there exists an infinitive set of machine indices $i_1, i_2 \ldots$ such that $L(M_{i_k}) = P$. Hence, the set $P \cap L$ is infinite.

Observe that for inputs $x \in P \cap L$ it must hold $x \in L(M_i)$ and hence $\mathrm{time}_{M_i} > 2T_1(|x|)$. Let $l := R_L(n_i) + 2^{2n_i+1}$. Then $l \leq 2^{2n_i+2}$ since $R_L(n_i) \leq n_i$.

The time complexity of M_i can be estimated by

$$\begin{aligned}
\sum_{\mathrm{rank}_L(x) \leq l} \frac{\mathrm{time}_{M_i}(x)}{T_1(|x|)} &\geq \sum_{R_L(n_i) \leq \mathrm{rank}_L(x) \leq R_L(n_i)+2^{2n_i+1}} \frac{2 \cdot T_1(|x|)}{T_1(|x|)} \\
&= \Big(R_L(n_i) + 2^{2n_i+1} - R_L(n_i) + 1\Big) \cdot 2 \\
&\geq 2^{2n_i+2} + 1 \ \geq \ l + 1 \ ,
\end{aligned}$$

contradicting the assumption that M_i were T_1–time bounded on the average. □

LEMMA 12. *Let $\delta > 1$, $V_2 \geq \mathcal{N}$, $V_2(\delta \cdot \mathcal{N}) \leq T_1$ and $T_1 \leq T_2(o(\mathcal{N}))$. Then for all infinite $L \in DTime(V_2)$ there exists a language $P \in DTime(T_2)$ such that*

$$P \ \cap \ L \ \notin \ \mathrm{AvTime}(T_1, \{rank_L\}) \ .$$

PROOF. Define P and the sequence n_i as above, but replace the condition $\frac{T_2(j)}{T_1(j)} \geq i$ by $\frac{T_2(j)}{T_1(2 \cdot j)} \geq i$. Modify the simulation in the proof of lemma 11 to testing whether a machine M_i does not accept an input x in time $T_1(2 \cdot |x|)$.

Remember that $T_1 \leq T_2(o(\mathcal{N}))$. So f is well-defined, and such simulations of M_i on input x can be performed in time $T_2(|x|)$. $P \in \mathrm{DTime}(T_2)$ follows as above.

$P \cap L \notin \mathrm{AvTime}(T_1, \{\mathrm{rank}_L\})$: Assume $P \cap L \in \mathrm{AvTime}(T_1, \mathrm{rank}_L)$ and let M_i be a DTM for this language with $(\mathrm{time}_{M_i}, \mathrm{rank}_L) \in \mathrm{Av}(T_1)$. Again letting $l := R_L(n_i) + 2^{2n_i+1}$ we get the contradiction

$$\begin{aligned}
\sum_{\mathrm{rank}_L(x) \leq l} \frac{{T_1}^{-1}\mathrm{time}_{M_i}(x)}{|x|} &\geq \sum_{R_L(n_i) \leq \mathrm{rank}_L(x) \leq R_L(n_i)+2^{2n_i+1}} \frac{{T_1}^{-1}(T_1(2 \cdot |x|)}{|x|} \\
&= \Big(R_L(n_i) + 2^{2n_i+1} - R_L(n_i) + 1\Big) \cdot 2 \\
&\geq 2^{2n_i+2} + 1 \ \geq \ l + 1 \ .
\end{aligned}$$

□

Combining the last four lemmas we get

THEOREM 7.1. *Let $\delta > 1$. Then for all bounds T, T', V_1, V_2 with $\mathcal{N} \leq V_1 \leq o(V_2)$, $V_2 \leq O(T)$ and $V_2(\delta \cdot \mathcal{N}) \leq O(T')$ holds*

$$\begin{aligned} \mathrm{EavTime}(T, V_2\text{–}\mathit{rankable}) &\subset \mathrm{EavTime}(T, V_1\text{–}\mathit{rankable}) \ , \\ \mathrm{AvTime}(T', V_2\text{–}\mathit{rankable}) &\subset \mathrm{AvTime}(T', V_1\text{–}\mathit{rankable}) \ . \end{aligned}$$

PROOF. Choose $T = T_1$ where T_1 is the complexity bound considered in the previous lemmas. The language $P \cap L$ constructed above in Lemmata 9 and 11 is contained in the average class EavTime(T_1, V_1–rankable) according to Lemma 9, but not in EavTime(T,V_2–rankable) according to Lemma 9 since $\mathrm{rank}_L \in V_2$–rankable (Lemma 9). The analogous property for the Av-measure holds for the language obtained from Lemmata 10 and 12 and $T' = T_1$. □

COROLLARY 9. *For $\mathcal{N} \leq V_1 \leq o(V_2)$, $V_1 \leq O(T)$ and $T \in POL$*

$$\mathrm{AvTime}(T, V_2\text{–}\mathit{rankable}) \subset \mathrm{AvTime}(T, V_1\text{–}\mathit{rankable}) \ .$$

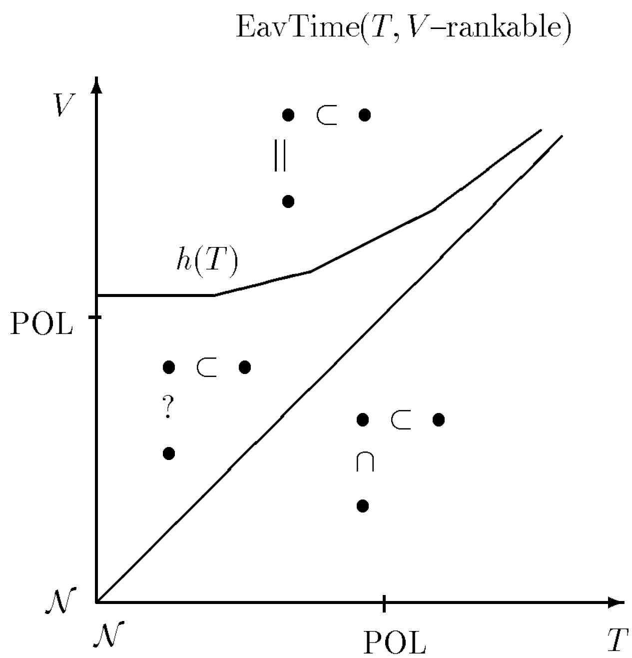

Figure 7.3: Hierarchies between expected average complexity classes EavTime(T)

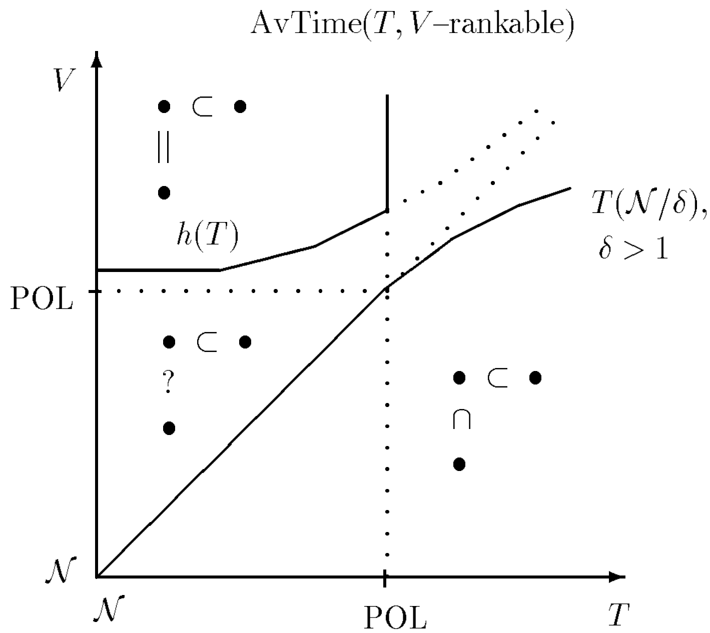

Figure 7.4: Hierarchies between average complexity classes AvTime(T)

Figures 7.3 and 7.4 give a pictorial description of the hierarchies implied by the last two theorems. Each point $\bullet$ in the diagram represents a complexity class EavTime(T, V–rankable), respectively AvTime(T, V–rankable). $\bullet \subset \bullet$ means strict inclusion between the two classes, $\bullet = \bullet$ equality and $\bullet \; ? \; \bullet$ that the relation cannot be deduced from the results obtained above.

8. Conclusions

We have shown that the average time complexity of an algorithm can be estimated as precisely as in the worst case. Ranking the input space and measuring the complexity of a distribution with respect to its rankability has turned out to be an appropriate and natural concept. Classical results like tight hierarchies can be obtained this way, both for the time complexity and the complexity of the distributions. Based on these notions, starting with distributional complexity classes we have presented meaningful definitions of average complexity classes the elements of which are languages in the standard sense. They are directly comparable to worst case classes.

These ideas can also be applied to other cases like the analysis of average space complexity.

References

S. Ben-David, B. Chor, O. Goldreich, and M. Luby, On the theory of average complexity. *J. CSS* **44** (92).

Y. Gurevich, Average completeness. *J. CSS* **42** (91), 346–398.

L. Levin, Average complete problems. *SIAM J. Computing* **15** (86), 285–286.

P. Miltersen, The complexity of malign ensembles. *SIAM J. Computing* **22** (93), 147–156.

D. Mitchell, B. Selman, and H. Levesque, Hard and easy distributions of SAT problems. In *Proc. 10. Nat. Conf. on Artificial Intelligence*, 92, 459–46.

R. Reischuk and C. Schindelhauer, Precise average complexity. In *Proc. 10. STACS*, 93, 650–661.

C. Schindelhauer, Neue Average Komplexitätsklassen. Master's thesis, Technische Hochschule Darmstadt, 91.

Rüdiger Reischuk
Institut für Theoretische Informatik
Medizinische Hochschule zu Lübeck
Wallstr. 40, D-23560 Lübeck
`reischuk@informatik.mu-luebeck.de`

Christian Schindelhauer
Institut für Theoretische Informatik
Medizinische Hochschule zu Lübeck
Wallstr. 40, D-23560 Lübeck
`schindel@informatik.mu-luebeck.de`