Electronic Colloquium on Computational Complexity, Report No. 46 (2003)

Locally Computed Baire's Categories on Small Complexity Classes

Philippe Moser*

Abstract

We strengthen the resource-bounded Baire's categories of [Mos03], and define resource bounded Baire's categories on small complexity classes such as P, QP, SUBEXP and on probabilistic complexity classes such as BPP. We give an alternative characterization of meager sets via resource-bounded Banach Mazur games. We show that the class SPARSE is meager in P. We investigate the genericity notion arising from our categories, and show that generic sets are REC-immune. We show that there is no weak-completeness notion based on our categories, i.e. every weakly-complete set is complete. Next we prove that the class of complete sets for P under Turing-logspace reductions is meager in P if P is not equal to DSPACE($\log n$), and that the same holds unconditionally for QP. Finally we show the following for our resource-bounded Baire's category on SUBEXP. First BPP is meager unless BPP equals EXP infinitely often. Second NP equals AM infinitely often unless NP is meager, and finally either RP is meager or ZPP equals RP infinitely often.

1 Introduction

Resource-bounded measure and resource-bounded Baire's Category were introduced by Lutz in [Lut90] and [Lut92] for both complexity classes E and EXP. Both provide a means of investigating the sizes of various subsets of E and EXP. In resource-bounded measure the small sets are those with measure zero, in resource-bounded Baire's Category the small sets are those of first category (meager sets). Both smallness notions satisfy the following three axioms. First every single language $L \in \mathsf{E}$ is small, second the whole class E is large, and finally "easy infinite unions" of small sets are small. These axioms meet the essence of both Lebegue's measure and Baire's category and ensure that it is impossible for a subset of E to be be both large and small.

The first goal of Lutz's approach was to extend existence results, such as "there is a language in C satisfying property P", to abundance results such as "most languages in C satisfy property P", which is more informative since an abundance result reflects the typical behavior of languages in a class, whereas an existence result could as well correspond to an exception in the class. Both resource-bounded measure and resource-bounded Baire's Category have been successfully used to understand the structure of the exponential time classes E and EXP.

An important question in resource-bounded measure theory was to generalize Lutz's measure theory to small complexity classes such as P, QP and SUBEXP and to probabilistic classes such

*Address: Computer Science Department, University of Geneva. Email: moser@cui.unige.ch

ISSN 1433-8092

as BPP and BPE. These issues have been solved in the following list of papers [AS94], [Str97], [RS98] and [Mos02].

The same question in the Baire's category setting was addressed in [AS95] , and resource-bounded Baire's categories on small and probabilistic classes were introduced in [Mos03]. In this paper we strengthen the resource-bounded Baire's categories of [Mos03] by introducing a much more powerful definition of resource-bounded Baire's category. To this end we consider locally computable strategies, as in [Fen95]. Informally speaking a class is said meager if there is a single strategy that defeats every language in the class. In [Mos03] the output of the strategy is required to be computable in polynomial time. For locally computable strategies, we only require the output to be bit-wise polynomially computable. Thus the output of locally computable strategies can be of any finite size. With this definition, we obtain a stronger resource-bounded Baire's category notion as in [Mos03], on complexity classes such as P, QP, SUBEXP and BPP. We give an alternative characterization of meager sets via resource-bounded Banach Mazur games. Next we show that the class SPARSE of languages with polynomial density is meager, in the sense of Baire's categories on P, which improves [Mos03] where SPARSE was not meager in the sense of categories on P.

Whereas typical sets for measure theory are called random sets, typical sets for Baire's category are called generic sets. Our category notions give rise to strong notions of genericity. We investigate the genericity concept yielded by our category notion on P. A natural question is whether there exists such a generic set in a class containing P such as E. We prove that this is not the case by showing that every such generic set is REC-immune.

In [Lut95] Lutz introduced the concept of weak completeness. A set A is said weakly complete if its lower span (the class of sets reducible to A) has measure non-zero. Lutz showed in [Lut95] the existence of weakly-complete sets that are not complete. Similarly we can define a categorical weak completeness notion, by calling a set A weakly complete if its lower span is not meager. We show that strictly weakly-complete languages do not exist, for our category notion on P, i.e. every weakly-complete language is also complete for P. Next we prove that the class of complete languages for P under Turing-logspace reduction is meager in P, if P is not equal to DSPACE($\log n$), and that the same holds unconditionally for QP.

In [Mel00] a zero-one law was proven for BPP, that is either BPP has measure zero in E or BPP = EXP. We show an analogue for categories on the smaller class SUBEXP, namely that either BPP is meager in SUBEXP, or BPP equals EXP infinitely often.

In [IM03] it is shown that under the plausible assumption "NP has measure non zero in E" total derandomization of NP is possible i.e. NP = AM. We show that under the plausible assumption "NP is not meager" for categories on the smaller class SUBEXP, partial derandomization of NP is achievable, i.e. NP equals AM infinitely often.

Finally we show that either RP small in or BPP is easy, that is either RP is meager in SUBEXP or ZPP equals BPP infinitely often.

2 Preliminaries

We use standard notation for traditional complexity classes; see for instance [BDG95], and [BDG90], or [Pap94]. For $\epsilon > 0$, denote by E_ϵ the class $\mathsf{E}_\epsilon = \bigcup_{\delta<\epsilon} \mathsf{DTIME}(2^{n^\delta})$. SUBEXP is

the class $\cap_{\epsilon>0}\mathsf{E}_\epsilon$, and quasi polynomial time refers to the class $\mathsf{QP} = \cup_{k\geq 1}\mathsf{DTIME}(n^{\log^k n})$, and $\mathsf{QP}_1 = \cup_{k\geq 1}\mathsf{DTIME}(n^{k\log n})$. Informally speaking QP_1 and QP share the same relationship as E and EXP. The point is that whereas it is easy to show that the canonical complete language is complete for QP_1, it is not easy to exhibit a complete language for QP. Let us fix some notations for strings and languages. Let $s_0, s_1, \ldots$ be the standard enumeration of the strings in $\{0,1\}^*$ in lexicographical order, where $s_0 = \lambda$ denotes the empty string. A sequence is an element of $\{0,1\}^\infty$. If w is a string or a sequence and $1 \leq i \leq |w|$ then $w[i]$ and $w[s_i]$ denotes the ith bit of w. Similarly $w[i\ldots j]$ and $w[s_i \ldots s_j]$ denote the ith through jth bits, and by $dom(w)$ the domain of w, where w is viewed as a partial function. We identify language L with its characteristic function χ_L, where χ_L is the sequence such that $\chi_L[i] = 1$ iff $s_i \in L$. For a string s_i define its position by $pos(s_i) = i$. If w_1 is a string and w_2 is a string or a sequence extending w_1, we write $w_1 \sqsubseteq w_2$. We write $w_1 \sqsubset w_2$ if $w_1 \sqsubseteq w_2$ and $w_1 \neq w_2$. For two strings $\tau, \sigma \in \{0,1\}^*$, we denote by $\tau^\frown\sigma$ or by $\tau\sigma$ the concatenation of τ followed by σ. For $a, b \in \mathbb{N}$ let $a \dot{-} b$ denote $\max(a-b, 0)$. We identify $\mathbb{N}$ with $\{0,1\}^*$, thus we denote by $\mathbb{N}^\mathbb{N}$ the set of all function mapping strings to strings.

2.1 Finite extension strategies

Whereas measure is defined via martingales, Baire's categories are defined via finite extension strategies. Here is a definition.

Definition 1 *A function $h : \{0,1\}^* \to \{0,1\}^*$ is a finite extension strategy, or a constructor, if for every string $\tau \in \{0,1\}^*$, $\tau \sqsubseteq h(\tau)$.*

For simplicity we will use the word "strategy" for finite extension strategy. We will often consider indexed strategies. An indexed strategy is a function $h : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$, such that $h_i := h(i, \cdot)$ is a strategy for every $i \in \mathbb{N}$. If h is a strategy and $\tau \in \{0,1\}^*$, define $ext\, h(\tau)$ to be the unique string u such that $h(\tau) = \tau^\frown u$. We say a strategy h avoids some language A (or language A avoids strategy h) if for every string $\tau \in \{0,1\}^*$ we have $h(\tau) \not\sqsubseteq \chi_A$. We say a strategy h meets some language A if h does not avoid A.

3 Local Categories on P

To define a resource bounded Baire's category on P, we shall consider strategies computed by Turing machines which have random access to their inputs, i.e. on input τ, the machine can query any bit of τ to its oracle. In order to allow such Turing machines to compute the lengths of their inputs τ without querying their oracles, we also provide them with $s_{|\tau|}$. For such a Turing machine M running on input τ, we denote this convention by $M^\tau(s_{|\tau|})$.

Let h be an indexed strategy. Consider the following function ext. Let $\sigma \in \{0,1\}^*$ and $i, k \in \mathbb{N}$ and let w be the unique string such that $h_i(\sigma) = \sigma^\frown w$. Define

$$ext(h_i(\sigma), k) = \begin{cases} w[k] & \text{if } 1 \leq k \leq |w| \\ \perp & \text{otherwise.} \end{cases} \qquad \text{and } ext(h_i(\sigma)) = w.$$

We shall consider strategies whose extensions are bit-wise computable in polynomial time. Such strategies are very strong since the extension can be of any finite size, as long as it is

locally computable. To guarantee that the second axiom holds, one needs to reduce the power of our strategies, by requiring that all queries made to the input are contained in a polynomial printable set, called the query set.

Definition 2 *An indexed strategy $h : \mathbb{N}\times\{0,1\}^* \to \{0,1\}^*$ is said $\mathsf{P}_{\mathsf{loc}}$ computable if there exists a random access Turing machine M as above such that for every $\tau \in \{0,1\}^*$ and every $i,k \in \mathbb{N}$,*

$$M^{\tau}(s_{|\tau|}, 0^i, k) = ext\,(h_i(\tau), k)$$

where M runs in time polynomial in $|s_{|\tau|}|+i+|k|$, and there is a poly printable query set G such that for every $n,i,k \in \mathbb{N}$ and for every $i',k' \in \mathbb{N}$ such that $i' \leq i$ and $k' \leq k$ and for every input $\sigma \in \{0,1\}^*$ such that $|s_{|\sigma|}| \leq n$, $M^{\sigma}(s_{|\sigma|}, 0^{i'}, k')$ queries σ only on bits that are in $G(0^n, 0^i, k)$, where $G(0^n, 0^i, k)$ is printable in time polynomial in $n+i+|k|$.

A class of languages is said meager if there is a single strategy that avoids every language in the class.

Definition 3 *A class C of languages is $\mathsf{P}_{\mathsf{loc}}$-meager if there exists a $\mathsf{P}_{\mathsf{loc}}$ computable indexed strategy h, such that for every $L \in C$ there exists $i \in \mathbb{N}$, such that h_i avoids L.*

In order to state the third axiom, one needs to define "easy infinite unions" precisely.

Definition 4 *$X = \bigcup_{i\in\mathbb{N}} X_i$ is a $\mathsf{P}_{\mathsf{loc}}$-union of $\mathsf{P}_{\mathsf{loc}}$-meager sets, if there exists an indexed $\mathsf{P}_{\mathsf{loc}}$-computable strategy $h : \mathbb{N}\times\mathbb{N}\times\{0,1\}^* \to \{0,1\}^*$, such that for every $i \in \mathbb{N}$, $h_{i,.}$ witnesses X_i's meagerness.*

3.1 Verifying the axioms

Let us check that all three axioms hold for our local categories.

Theorem 1 *For any language L in P, the singleton $\{L\}$ is $\mathsf{P}_{\mathsf{loc}}$-meager.*

Proof. Let $L \in \mathsf{P}$ be any language. We describe a $\mathsf{P}_{\mathsf{loc}}$-computable constructor h which avoids $\{L\}$. Consider the following Turing machine M computing h. For a string $\sigma \in \{0,1\}^*$, $M^{\sigma}(s_{|\sigma|}, k)$ simply outputs $1 - L(s_{|\sigma|+1})$ if $k = 1$ and $\perp$ if $k > 1$. Since M doesn't query its oracle it is easy to check that h is $\mathsf{P}_{\mathsf{loc}}$-computable. Let's check that h avoids $\{L\}$. Let σ be any prefix of χ_L, we have $h(\sigma) \not\sqsubseteq \chi_L$, i.e. h avoids χ_L. □

The following result states that subsets of small sets are small, and that easy infinite unions of small sets are small.

Theorem 2

1. *All subsets of a $\mathsf{P}_{\mathsf{loc}}$-meager set are $\mathsf{P}_{\mathsf{loc}}$-meager.*
2. *A $\mathsf{P}_{\mathsf{loc}}$-union of $\mathsf{P}_{\mathsf{loc}}$-meager sets is $\mathsf{P}_{\mathsf{loc}}$-meager.*

Proof. Immediate by definition of $\mathsf{P}_{\mathsf{loc}}$-meagerness. □

Let us prove the second axiom.

Theorem 3 P *is not* $\mathsf{P}_{\mathsf{loc}}$*-meager.*

Proof. We will need the following technical Lemma.

Lemma 1 *Let h be a $\mathsf{P}_{\mathsf{loc}}$ computable indexed strategy. Then there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that,*

1. *For every $\sigma \in \{0,1\}^*$ such that $|\sigma| \leq \sum_{j=0}^{i-1} f(j)$ we have $|ext\, h_i(\sigma)| \leq f(i)$,*
2. *$f(0) = 1$ and $f(i) \geq 2^i$ for every $i \in \mathbb{N}$,*

and there exists a deterministic Turing machine which on input i, computes $f(i)$ within $O(\log(f(i)))$ steps.

Proof. Let h be any $\mathsf{P}_{\mathsf{loc}}$ computable indexed strategy and let N be a Turing machine witnessing this fact. We construct a deterministic Turing machine M for f. At each computation's step, M increments a counter R. On input i, M computes f recursively as follows: $f(0) = 1$. For $i > 0$ compute $B := \sum_{j=0}^{i-1} f(j)$. For every string σ of size at most B, simulate $N^\sigma(s_{|\sigma|}, 0^i, k)$ for $k = 1, 2, \ldots$ until N outputs $\perp$, store under k_σ the corresponding k. Then compute $K := \max_{|\sigma| \leq B} k_\sigma$. Stop incrementing the counter R, compute 2^{R+K+i}, and output this value.

For the running time of M on input i, observe that the last two steps (once the counter R is stopped) take time $O(R + K + i)$. Thus the total running time of M is at most $O(R + K + i)$ which is less than $O(\log(f(i)))$. This ends the proof of the Lemma.

Let us prove the Theorem. Let h be an indexed constructor in $\mathsf{P}_{\mathsf{loc}}$ and let M be a Turing machine computing h with query set G_M. Let f be as in Lemma 1 and let Q be a Turing machine computing f.

We construct a language $L \in \mathsf{P}$ which meets h_i for every i. The idea is to construct a language L with the following characteristic function,

$$\chi_L = |\underbrace{0}_{B_0}|\underbrace{ext\, h_1(B_0)0\cdots0}_{B_1}|\underbrace{ext\, h_2(B_0 \hat{} B_1)0\cdots0}_{B_2}|\cdots|\underbrace{ext\, h_i(B_0 \hat{} B_1 \hat{} \cdots \hat{} B_{i-1})0\cdots0}_{B_i}|$$

where block B_i has size $f(i)$ and contains $ext\,(h_i(B_0 \hat{} B_1 \hat{} \cdots \hat{} B_{i-1}))$ followed by a padding with 0's. B_i is large enough to contain $ext\,(h_i(B_0 \hat{} B_1 \hat{} \cdots \hat{} B_{i-1}))$ by definition of f.

Let us construct a polynomial time Turing machine N deciding L. On input x, where $|x| = n$,

1. Compute $pos(x)$.
2. Compute the index $i(x)$, where the membership bit of x is in zone $B_{i(x)}$, with the formula $i(x) = \max_{j \geq 0}[\sum_{t=0}^{j-1} f(t) < pos(x) + 1]$, in the following way. At the beginning $S = 1$, then for $t = 1, 2, \ldots$ compute $f(t)$ by simulating Q for $|x|^2$ steps, and add the result to S, until either Q doesn't halt, or $S \geq pos(x) + 1$. Let t_0 denote the first t for which this happens, then $i(x) = t_0 - 1$.
3. Compute the position of x in $B_{i(x)}$, where $rpos(x) = pos(x) - F(i(x) - 1)$, with $F(j) = \sum_{t=0}^{j} f(t)$.

4. Compute the membership bit of x, where $bit(x) = ext\,(h_{i(x)}(B_0 \wedge B_1 \wedge \cdots \wedge B_{i(x)-1}), rpos(x))$. If $bit(x) = \perp$, then output 0 (x is in the padded zone of $B_{i(x)}$), otherwise output $bit(x)$.

Let us check that L is in P. The first step is clearly computable in time polynomial in n. For the second step notice that if $f(t) < pos(x)$, Q must halt within $O(\log(pos(x)))$ steps, which is less than $|x|^2$ since $pos(x) = 2^{O(|x|)}$. Thus the second step computes $i(x)$ correctly. Moreover since f increases at least exponentially, only a polynomial number of terms need to be summed in the second and third step. Since f is at least exponentially increasing, the sums in step two and three can be done in polynomial time. Finally the last step requires simulating $M^{B_0 \wedge B_1 \wedge \cdots \wedge B_{i(x)-1}}(s_{F(i(x)-1)}, 0^{i(x)}, rpos(x))$. By the hypothesis on h, M's queries are all in $G_M(0^{|s_{F(i(x)-1)}|}, 0^{i(x)}, rpos(x))$, which is contained in $G_M(0^{|s_{F(i(x)-1)}|}, 0^{i(x)}, pos(x))$ which has size polynomial in $|x|$. For such a query q, i.e. M queries the qth bit of its input, simply run step one to four above with x replaced by q. By definition of G_M only a polynomial number of recursive steps need to be performed. □

3.2 Resource-bounded Banach-Mazur games

We give an alternative characterization of small sets via resource-bounded Banach-Mazur games. Informally speaking, a Banach-Mazur game, is a game between two strategies f and g, where the game begins with the empty string on which $g \circ f$ is applied successively. Such a game yields a unique infinite string, i.e. a language, called the result of the play between f and g. For a class C, we say that g is a winning strategy if it can force the result of the game with any strategy f to be a language not in C. We show that the existence of a winning strategy is equivalent to the meagerness of C. This equivalence result is useful in practice, since it is often easier to find a winning strategy, rather than a finite extension strategy.

Definition 5

1. *A play of a Banach-Mazur game is a pair (f, g) of strategies such that for every string $\tau \in \{0,1\}^*$, $\tau \sqsubset g(\tau)$.*

2. *The result $R(f, g)$ of the play (f, g) is the unique element of $\{0,1\}^\infty$ that extends $(g \circ f)^i(\lambda)$ for every $i \in \mathbb{N}$.*

For a class of languages C and two function classes F_I and F_{II}, denote by $G[C, F_I, F_{II}]$ the Banach-Mazur game with distinguished set C, where player I must choose a strategy in F_I, and player II a strategy in F_{II}. We say player II wins the play (f, g) if $R(f, g) \notin C$, otherwise we say player I wins. We say player II has a winning strategy for the game $G[C, F_I, F_{II}]$, if there exists a strategy $g \in F_{II}$ such that for every strategy $f \in F_I$, player II wins (f, g)

Let us prove that both notion are equivalent.

Theorem 4 *Let X be any class of languages. The following are equivalent.*

1. *Player II has a winning strategy for $G[X, \mathbb{N}^{\mathbb{N}}, \mathsf{P}_{\mathsf{loc}}]$.*

2. *X is $\mathsf{P}_{\mathsf{loc}}$-meager.*

Proof. We need the following technical Lemma.

Lemma 2 *Let h be a $\mathsf{P}_{\mathsf{loc}}$ computable indexed constructor. Then there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that for every $m \in \mathbb{N}$, $t \leq m$ and for every string τ of size at most m, $|h_t(\tau)| \leq f(m)$, and there exists a deterministic Turing machine which on input m, computes $f(m)$ within $O(f(m))$ steps.*

Proof. Let h be a $\mathsf{P}_{\mathsf{loc}}$ computable strategy, and let N be a Turing machine witnessing this fact. We construct a deterministic Turing machine computing f. At each computation's step, M increments a counter R. On input $m \in \mathbb{N}$, compute $K = max_{|\tau| \leq m, t \leq m}\{|h_t(\tau)|\}$, by simulating N on all appropriate strings. Stop incrementing the counter R, compute $K + R$ and output the result. Thus M's total running time is less than $O(R + K)$ which is less than $O(f(m))$. This ends the proof of the Lemma.

For the proof of the Theorem, suppose the first statement holds and let $g \in \mathsf{P}_{\mathsf{loc}}$ be a wining strategy for player II. Let M be a Turing machine computing g. We define an indexed constructor $h \in \mathsf{P}_{\mathsf{loc}}$ by constructing a machine N for h; let $i \in \mathbb{N}$ and $\sigma \in \{0,1\}^*$,

$$h_i(\sigma) := g(\sigma') \quad \text{where } \sigma' = \sigma \,\hat{}\, 0^{\dot{i-|\sigma|}}.$$

h is $\mathsf{P}_{\mathsf{loc}}$-computable because computing $h_i(\sigma)$ simply requires to simulate $M^{\sigma'}(s_{|\sigma'|})$ answering M's queries in $dom(\sigma')\backslash dom(\sigma)$ by 0. Thus N's query set satisfies $G_N(0^{|s_{|\sigma|}|}, 0^i, k) \subseteq G_M(0^{|s_{|\sigma'|}|}, k)$ which has size polynomial in $|s_{|\sigma|}| + i + |k|$, because $|\sigma'| \leq |\sigma| + i$.

We show that if language A meets h_k for every $k \in \mathbb{N}$, then $A \notin X$. This implies that X is $\mathsf{P}_{\mathsf{loc}}$-meager as witnessed by h. To do this we show that for every $\alpha \sqsubset \chi_A$ there is a string β such that,

$$\alpha \sqsubseteq \beta \sqsubseteq g(\beta) \sqsubset \chi_A.$$

If this holds, then player I has a winning strategy yielding $R(f, g) = A$: for a given α player I extends it to obtain the corresponding β, thus forcing player II to extend to a prefix of χ_A. So let α be any prefix of χ_A, where $|\alpha| = k$. Since A meets h_k, there is a string $\sigma \sqsubset \chi_A$ such that

$$\sigma' \sqsubseteq g(\sigma') = h_k(\sigma) \sqsubset \chi_A$$

where $\sigma' = \sigma \,\hat{}\, 0^{\dot{k-|\sigma|}}$. Since $|\alpha| \leq |\sigma'|$ and α, σ' are prefixes of χ_A, we have $\alpha \sqsubseteq \sigma'$. Define β to be σ'.

For the other direction, let X be $\mathsf{P}_{\mathsf{loc}}$-meager as witnessed by h, i.e. for every $A \in X$ there exists $i \in \mathbb{N}$ such that h_i avoids A. Let N be a Turing machine computing h. Let $f : \mathbb{N} \to \mathbb{N}$ be as in Lemma 2, and let Q be a deterministic Turing machine computing f. We define a constructor $g \in \mathsf{P}_{\mathsf{loc}}$ inducing a winning strategy for player II in the game $G[X, \mathbb{N}^{\mathbb{N}}, \mathsf{P}_{\mathsf{loc}}]$. We show that for any strategy f, $R(f, g)$ meets h_i for every $i \in \mathbb{N}$, which implies $R(f, g) \notin X$. Here is a description of a Turing machine M computing g. For a string σ with $|s_{|\sigma|}| = n$, $M^{\sigma}(s_{|\sigma|}, k)$ does the following.

1. Compute $B = \max_{m \geq 1}[f(m) \leq n]$ in the following way. For $t = 1, 2, \ldots$ compute $f(t)$ by simulating Q for n^2 steps, and denote the result by b_t, until either Q doesn't halt, or $b_t > n$. Let t_0 denote the first t for which this happens, then define $B = b_{t_0 - 1}$.

2. Compute $n_0 = \min_{t \leq B}[(\forall \tau \sqsubseteq \sigma \text{ such that } |\tau| \leq B) \;\; h_t(\tau) \not\sqsubseteq \sigma]$.

3. If no such n_0 exists output 0 if $k = 1$, and output $\perp$ if $k > 1$.

4. If n_0 exists, then if $k = 1$ output 0, otherwise simulate $N^{\sigma \hat{} 0}(s_{|\sigma|+1}, 0^{n_0}, k-1)$ answering N's queries in $dom(\sigma \hat{} 0) \backslash dom(\sigma)$ with 0, and output the result of the simulation.

Let us check that g is $\mathsf{P_{loc}}$ computable. For the first step, we have that whenever $f(m) \leq n$ Q halts within $O(n)$ steps. Since Q is simulated n^2 steps, B is computed correctly, in polynomial time. For the second step, the B^3 simulations of N can be done in polynomial time. Moreover every $h_t(\tau)$ computed during the second step has size at most B, thus only the first n bits of the input σ need to be read. This together with the fourth step guarantees that the query set for M is given by $G_M(0^{|s_{|\sigma|}|}, k) = \{1, 2, \ldots n\} \bigcup G_N(0^{|s_{|\sigma|+1}|}, 0^n, k-1)$ which has polynomial size.

We show that $R(f, g)$ meets every h_i for any strategy f. Indeed suppose this is not the case, i.e. there is a strategy f such that $R(f, g)$ does not meet h. Let n_0 be the smallest index such that $R(f, g)$ does not meet h_{n_0}. Since $R(f, g)$ meets h_{n_0-1} there is a string τ such that $h_{n_0-1}(\tau) \sqsubset R(f, g)$. Since g strictly extends strings at every round, after a certain number of rounds, f will output a string σ long enough to enable step 2 (of M's description) to find out that $h_{n_0-1}(\tau) \sqsubseteq \sigma$ thus incrementing $n_0 - 1$ to n_0. At this round we have $g(\sigma) = \sigma \hat{} 0 \hat{} ext\, h_{n_0}(\sigma \hat{} 0)$, i.e. $h_{n_0} \sqsubset R(f, g)$ which is a contradiction. □

Note that throughout Section 3, the polynomial time bounds in $\mathsf{P_{loc}}$ categories, can be replaced by both quasipolynomial and subexponential time bounds, thus yielding a local category notion on both complexity classes QP and SUBEXP. In section 6 and 8, some applications of our local category notion on QP and SUBEXP will be given.

4 Meagerness of SPARSE

It was shown in [Mos03] that the class SPARSE is not meager for the category notion on P introduced in [Mos03]. Here we prove that local computable strategies are stronger than the strategies of [Mos03], by showing that the class SPARSE is $\mathsf{P_{loc}}$-meager.

Theorem 5 SPARSE *is* $\mathsf{P_{loc}}$*-meager.*

Proof. Let L be any sparse language. Then there exists a polynomial p such that $|L \cap \{0,1\}^n| \leq p(n)$, for every $n \geq 1$. Consider the following strategy h, which on input $\sigma \in \{0,1\}^*$ pads σ with $|\sigma|$ 1's. Since L is sparse, h avoids L. We construct a random access Turing machine M for h; on input $\sigma \in \{0,1\}^*$ and $j \in \mathbb{N}$, $M^{\sigma}(s_{|\sigma|}, j)$ outputs 1 if $1 \leq j \leq |\sigma|$ and $\perp$ otherwise. Since M doesn't query its oracle, h is $\mathsf{P_{loc}}$-computable which ends the proof. □

5 Genericity

The typical sets for Baire's category are generic sets. Here is a definition.

Definition 6 *A language G is* $\mathsf{P_{loc}}$*-generic if G meets every* $\mathsf{P_{loc}}$*-computable strategy.*

The following result shows that the genericity notion yielded by our local categories on P is a very strong one: every generic set is REC-immune.

Theorem 6 *Let G be a $\mathsf{P}_{\mathsf{loc}}$-generic set. Then G is REC-immune.*

Proof. We need the following technical Lemma.

Lemma 3 *Let A be any recursively enumerable set. Then there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that range$(f) = A$, i.e. range$(f) = \{N | s_N \in A\}$, and $f(n)$ is computable in time $O(n)$.*

Proof. Let M be a Turing machine for A. To compute $f(n)$, simulate M during a total of n steps on $s_1, s_2, \cdots$. Denote by s_i, if it exists, the last string on which M stops and outputs 1 . Define $f(n) = i$ if s_i exists, otherwise $f(n) = 0$.

This ends the proof of the Lemma. For the proof of the Theorem, let G be a $\mathsf{P}_{\mathsf{loc}}$-generic set, then G is infinite because G meets every strategy $s_k(\sigma) = \sigma \,^\wedge \underbrace{1 \,^\wedge \ldots \,^\wedge 1}_{k}$. Suppose for a contradiction that there exists an infinite recursively enumerable set $A \subseteq G$, and let f be as in Lemma 3. Consider the following $\mathsf{P}_{\mathsf{loc}}$-computable strategy h.

$$ext(h_i(\sigma), k)) = \begin{cases} 0 & \text{if } \{f(0), f(1), \ldots, f(|k|)\} \cap \{|\sigma|, |\sigma|+1, \ldots, |\sigma|+|k|-1\} = \emptyset \\ \perp & \text{otherwise.} \end{cases}$$

Since A is infinite, there exists n_0 such that $\{f(0), f(1), \ldots, f(n_0)\} \cap \{|\sigma|, |\sigma|+1, \ldots, |\sigma|+n_0-1\}$ is non-empty, and since this set can only increase in size with k, we have $ext(h_i(\sigma), k)) = \perp$ for every k with $k \geq n_0$. moreover h is computable in polynomial time, thus h is $\mathsf{P}_{\mathsf{loc}}$-computable. Since h extends characteristic strings of languages with 0's up to the membership bit of a word x such that $A(x) = 1$, h avoids A. Since $A \subseteq G$, h avoids G, which ends the proof. □

6 Weak Completeness

In [Lut95] the concept of weak completeness was introduced. A set A is said weakly complete if its lower span (the class of sets reducible to A) has measure non-zero. Lutz showed in [Lut95] the existence of weakly-complete sets that are not complete. Similarly we can define a categorical weak completeness notion, by calling a set A weakly complete if its lower span is not meager. We show that there is no weak-complete but incomplete language, for our category notion on P, i.e. every weakly complete language for P is complete for P. To this end we need the following result.

Theorem 7 *Let C be a Σ_2^0 class, such that there exists a language A in P, such that for every finite variant A' of A, $A' \notin C$. Then C is $\mathsf{P}_{\mathsf{loc}}$-meager.*

Proof. By hypothesis there exists a polynomial oracle Turing machine M, such that $C = \{L | \exists x \forall y : M^L(x, y) = 0\}$. Consider the following $\mathsf{P}_{\mathsf{loc}}$-computable strategy g where $ext(g(\sigma), k))$, with $|s_{|\sigma|}| = n$, is computed as follows.

1. Simulate $M^L(x, y)$ for every $x < \log n$ and $y < \log k$, where $\chi_L = \sigma \,^\wedge A(s_{|\sigma+1|}) \,^\wedge A(s_{|\sigma+2|}) \cdots$.

2. If for every $x < \log n$ there exists $y < \log k$ such that $M^L(x,y) \neq 0$ output $\perp$, else output $A(s_{|\sigma|+k})$.

Let us show that g is a strategy. Suppose for a contradiction that g extends σ infinitely. Then the result is a finite variant of A, hence not in C. Therefore there exists $k \in \mathbb{N}$, such that $\forall x < |s_\sigma|\ \exists y < \log k$, such that $M^L(x,y) \neq 0$, where $L = g(\sigma)$. Hence g should not have extended σ more than k bits, which is a contradiction.

g is computable in polynomial time since there are a $\log n \cdot \log k$ simulations to perform and since the queries to A can be computed in polynomial time. The query set $G_g(n,k)$ has size polynomial in n and $|k|$, therefore g is $\mathsf{P}_{\mathsf{loc}}$-computable.

Let us show that g avoids C. Denote by L the result of the game between g and some strategy f, where player II plays according to g. Let z be any string. On the first turn for player II where the state of the game is of length at least $2^{2^{z+1}}$, player II will extend ensuring that $\forall x < z+1\ \exists y < \log k$ such that $M^L(x,y) \neq 0$. Thus $M^L(z,y) \neq 0$, which implies $L \notin C$. □

Corollary 1 *Let C be a Σ^0_2 class closed under finite variants. Then C is $\mathsf{P}_{\mathsf{loc}}$-meager iff $\mathsf{P} \not\subseteq C$.*

An interesting consequence of Corollary 1 is that every weakly-complete set is also complete.

Theorem 8 *There is no weakly-complete incomplete set for P under Turing-logspace reductions.*

Proof. Let $A \in \mathsf{P}$ be any language. Denote by $P(A)$ the lower span of A, i.e. the class of languages $\leq^T_{logspace}$-reducible to A. It is easy to check that $P(A)$ is a Σ^0_2 class and is closed under finite variant. Thus by Corollary 1 we have $P(A)$ is not $\mathsf{P}_{\mathsf{loc}}$-meager iff A is $\leq^T_{logspace}$-hard for P. □

Another consequence of Corollary 1 is the meagerness of the class of complete sets for P, under the assumption P is not equal to $\mathsf{DSPACE}(\log n)$.

Theorem 9 *If P is not equal to $\mathsf{DSPACE}(\log n)$, then the class of $\leq^T_{logspace}$-complete sets for P is $\mathsf{P}_{\mathsf{loc}}$-meager.*

Proof. Let A be a $\leq^T_{logspace}$-complete set for P. Consider the upper span of A denoted $P^{-1}(A)$, where $P^{-1}(A) = \{B : A \leq^T_{logspace} B\}$. It is easy to check that $P^{-1}(A)$ verifies the hypothesis of Corollary 1. Since $\mathsf{P} \not\subseteq P^{-1}(A)$, $P^{-1}(A)$ is $\mathsf{P}_{\mathsf{loc}}$-meager. Since every $\leq^T_{logspace}$-complete language for P is in $P^{-1}(A)$, the proof is complete. □

Note that the same result holds unconditionally for our category notion on QP_1, more precisely.

Theorem 10 *There is no weakly-complete incomplete set for QP_1 under Turing-logspace reductions.*

Proof. The proof is similar to Theorem 9.

7 Local Categories on BPP

In this section we generalize our local categories on P to the probabilistic class BPP. To this end we need the following probabilistic strategies.

Definition 7 *An indexed strategy $h : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ is said $\mathsf{BPP}_{\mathsf{loc}}$ computable if there is a probabilistic random access Turing machine (as defined in Section 3) M such that for every $\tau \in \{0,1\}^*$ and every $i,k,n \in \mathbb{N}$,*

$$\Pr[M^{\tau}(s_{|\tau|}, 0^i, k, 0^n) = ext\,(h_i(\tau), k)] \geq 1 - 2^{-n}$$

where the probability is taken over the internal coin tosses of M, M runs in time polynomial in $|s_{|\tau|}| + i + |k| + n$, and there is a poly printable query set G such that for every $m,i,k \in \mathbb{N}$ and for every $i',k' \in \mathbb{N}$ such that $i' \leq i$ and $k' \leq k$ and for every input $\sigma \in \{0,1\}^*$ such that $|s_{|\sigma|}| \leq m$, $M^{\sigma}(s_{|\sigma|}, 0^i, k, 0^n)$ queries σ only on bits that are in $G(0^m, 0^i, k)$, where $G(0^m, 0^i, k)$ is printable in time polynomial in $m + i + |k|$.

By using standard Chernoff bound arguments it is easy to show that Definition 7 is robust, i.e. the error probability can range from $1/2 + 1/p(n)$ to $1 - 2^{-q(n)}$ for any polynomials p, q, without enlarging (resp. reducing) the class of strategies defined in Definition 7.

Similarly to P, a class C is said meager if there is a single probabilistic strategy that avoids C.

Definition 8 *A class of languages C is $\mathsf{BPP}_{\mathsf{loc}}$-meager if there exists a $\mathsf{BPP}_{\mathsf{loc}}$ computable indexed strategy h, such that for every $L \in C$ there exists $i \in \mathbb{N}$, such that h_i avoids L.*

The definition of easy infinite unions is similar to P.

Definition 9 *$X = \bigcup_{i \in \mathbb{N}} X_i$ is a $\mathsf{BPP}_{\mathsf{loc}}$-union of $\mathsf{BPP}_{\mathsf{loc}}$-meager sets if there exists an indexed $\mathsf{BPP}_{\mathsf{loc}}$-computable strategy $h : \mathbb{N} \times \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ such that for every $i \in \mathbb{N}$, $h_{i,\cdot}$ witnesses X_i's meagerness.*

7.1 Verifying the axioms

Let us prove the first axiom.

Theorem 11 *For any language L in $\mathsf{BPP}_{\mathsf{loc}}$, the singleton $\{L\}$ is $\mathsf{BPP}_{\mathsf{loc}}$-meager.*

Proof. Let $L \in \mathsf{BPP}$ be any language and let N be a probabilistic Turing machine deciding it. We describe a $\mathsf{BPP}_{\mathsf{loc}}$-computable constructor h which avoids $\{L\}$. Consider the following Turing probabilistic Turing machine M computing h. On input a string $\sigma \in \{0,1\}^*$, $M^{\sigma}(s_{|\sigma|}, k, 0^n)$ simulates $N(s_{|\sigma|+1})$ with error probability smaller than 2^{-n} and outputs $1 - N(s_{|\sigma|+1})$ if $k = 1$ and $\perp$ if $k > 1$. Since M doesn't query its oracle, it is easy to check that h is $\mathsf{BPP}_{\mathsf{loc}}$-computable, moreover h avoids $\{L\}$. □

The third axiom holds by definition.

Theorem 12

1. *All subsets of a* $\mathsf{BPP}_{\mathsf{loc}}$*-meager set are* $\mathsf{BPP}_{\mathsf{loc}}$*-meager.*

2. *A* $\mathsf{BPP}_{\mathsf{loc}}$*-union of* $\mathsf{BPP}_{\mathsf{loc}}$*-meager sets is* $\mathsf{BPP}_{\mathsf{loc}}$*-meager.*

Proof. Immediate by definition of $\mathsf{BPP}_{\mathsf{loc}}$-meagerness. □

Let us prove the second axiom.

Theorem 13 BPP *is not* $\mathsf{BPP}_{\mathsf{loc}}$*-meager.*

Proof. The proof is very similar to Theorem 3. Notice that only step 4 in the proof of Theorem 3 is probabilistic; all simulations of M in step 4 can be performed with error probability 2^{-n}. Since there is a polynomial number of simulations, L is in BPP. □

Similarly to P, meagerness can be characterized through Banach-Mazur games.

Theorem 14 *Let X be any class of languages. The following are equivalent.*

1. *Player II has a winning strategy for* $G[X, \mathbb{N}^{\mathbb{N}}, \mathsf{BPP}_{\mathsf{loc}}]$.

2. *X is* $\mathsf{BPP}_{\mathsf{loc}}$*-meager.*

Proof. The proof is similar to that of Theorem 4. For the first direction, M can be simulated with error probability 2^{-n}, thus h is in $\mathsf{BPP}_{\mathsf{loc}}$. For the other direction, less than n^2 simulation of h are performed in step 2, and each can be performed with error probability smaller than $2^{-s(n')}$ (where s is a polynomial to be determined later). Step 4 can be performed with error probability smaller than $2^{-s(n')}$; this yields a total error probability smaller than $(n^2+1)2^{-s(n')}$, which is smaller than $2^{-n'}$ for an appropriate choice of s. □

8 Consequences of some non-meagerness assumptions

In the following, we will show an application of our category notion on SUBEXP. It is easy to check that the category notion of Section 3, can be adapted to the class E_α, for every $\alpha > 0$. Let us denote by $\mathsf{E}_{\alpha_{\mathsf{loc}}}$ the corresponding category notion.

We show an analogue of the zero-one law for BPP [Mel00] for local categories on SUBEXP. We show that for every $\alpha > 0$, either BPP is $\mathsf{E}_{\alpha_{\mathsf{loc}}}$-meager or BPP is equal to EXP infinitely often.

Theorem 15 *For every $\alpha > 0$,* BPP *is* $\mathsf{E}_{\alpha_{\mathsf{loc}}}$ *-meager, unless* $\mathsf{BPP} \stackrel{i.o.}{=} \mathsf{EXP}$.

Proof. We need the following result.

Theorem 16 *Let C be a complexity class such that for every $\epsilon > 0$ and every language $A \in C$, there exists a language $B \in \mathsf{DTIME}(2^{n^\epsilon})$, such that for almost every $n \in \mathbb{N}$, $A(1^n) = B(1^n)$. Then C is* $\mathsf{E}_{\alpha_{\mathsf{loc}}}$*-meager, for every $\alpha > 0$.*

Proof. Let $\alpha > 0$. Let $M_1, M_2, \ldots$ be a standard enumeration of $\mathsf{DTIME}(2^{n^\delta})$, where δ will be determined later and where M_i runs in time $2^{n^\delta} + q(i)$ for some polynomial q, obtained by adding an alarm clock. Consider the following deterministic Turing machine M. On input 1^j,

M simulates $M_i(1^j)$ and outputs $1 - M_i(1^j)$, where i is computed as follows.

i	1	1	2	1	2	3	1	2	3	4	...
j	1	2	3	4	5	6	7	8	9	10	...

(1)

Let $i \in \mathbb{N}^*$ and denote by i_0 the first j for which i occurs in Table 1. We have $M(1^{i_0}) = 1 - M_i(1^{i_0})$, and $M(1^{i_0+t(i+1)-1}) = 1 - M_i(1^{i_0+t(i+1)-1})$ for every $t \geq 1$. Define M to output 0 on all other strings. Let $k \in \mathbb{N}^*$ and denote by T_k the set of languages L such that for almost every $n \in \mathbb{N}$, $L(1^n) = B_k(1^n)$, where $B_k = \mathcal{L}(M_k)$, and let $X = \bigcup_{k \in \mathbb{N}^*} T_k$. Consider the following strategy g. On input σ where $|s_{|\sigma|}| = n$, g is defined as follows,

$$ext\,(g(\sigma), k)) = \begin{cases} M(s_{|\sigma|+k}) & \text{if } 1 \leq k \leq 2^{5n} \\ \perp & \text{otherwise.} \end{cases}$$

Let us show that g is computable in time $2^{n^{\alpha/2}}$. On input σ and k as above, a machine for g needs to perform one simulation of M on an input less than 1^{5n}, i.e. simulate $M_i(1^j)$ with $i, j \leq 5n$. This takes time $2^{5n^\delta} + q(5n)$, which is less than $2^{n^{\alpha/2}}$ for an appropriate choice of δ.

We show that for every strategy f we have $R(f, g) \notin X$. Suppose for a contradiction that there is a strategy f such that $R(f, g) \in X$. Denote $L = R(f, g)$. Applying the hypothesis with $\epsilon = \delta$ yields $L \in T_i$ for some integer i, i.e. there exists $N_0 \in \mathbb{N}$ such that for every $s > N_0$, $L(1^s) = B_i(1^s)$. Let i_0 be as above, and let σ be a string output by f during the game between f and g, such that $|s_{|\sigma|}| = n$ with $n > i_0$ and $n > N_0$. Let t be the greatest integer such that $i_0 + t(i+1) - 1 < n + 1$. By definition of n we have,

$$n + 1 \leq \underbrace{i_0 + (t+1)i + t}_{I} < 4n.$$

Since $L = R(f, g)$, we have $L(1^I) = M(1^I) = 1 - M_i(1^I) = 1 - B_i(1^I)$ which is a contradiction because $I > N_0$. Since $C \subseteq X$, C is $\mathsf{E}_{\alpha_{\mathsf{loc}}}$-meager.

This ends the proof of Theorem 16. To prove Theorem 15 we also need the following slightly modified result of [IW98].

Theorem 17 (Impagliazzo, Widgerson) *If* $\mathsf{BPP} \overset{i.o.}{\neq} \mathsf{EXP}$, *then for every language* $A \in \mathsf{BPP}$ *and every* $\epsilon > 0$, *there is a language* $B \in \mathsf{DTIME}(2^{n^\epsilon})$ *such that the following holds: For any constant* $d > 0$ *and any length-preserving randomized polynomial-time Turing machine* M,

$$\Pr[A(M(1^m)) = B(M(1^m))] > 1 - m^{-d},$$

for almost every lengths m, *where the probability is taken over the internal coin tosses of* M.

Suppose $\mathsf{BPP} \overset{i.o.}{\neq} \mathsf{EXP}$. Applying Theorem 17 with $M(1^m) = 1^m$, we have that for every $A \in \mathsf{BPP}$ and every $\epsilon > 0$, there exists $B \in \mathsf{DTIME}(2^{n^\epsilon})$ such that $A(1^m) = B(1^m)$, for almost every lengths m. Thus by Theorem 16 we have that BPP is $\mathsf{E}_{\alpha_{\mathsf{loc}}}$-meager for every $\alpha > 0$. □

8.1 Partial Derandomization of AM under non-meagerness assumption

The following result achieves partial derandomization for NP under the hypothesis "NP is not meager" for categories on SUBEXP.

Theorem 18 $\mathsf{NP} \overset{i.o.}{=} \mathsf{AM}$, *unless* NP *is* $\mathsf{E}_{\alpha|\mathsf{oc}}$*-meager, for every* $\alpha > 0$.

Proof. The proof relies on Theorem 16 combined with the following result from [Lu00].

Theorem 19 *[Lu00] If* $\mathsf{NP} \overset{i.o.}{\neq} \mathsf{AM}$, *then for every language* $A \in \mathsf{NP}$ *and every* $\epsilon > 0$, *there is a language* $B \in \mathsf{DTIME}(2^{n^\epsilon})$ *such that the following holds: For any length-preserving nondeterministic polynomial-time multi-valued transducer* M, *there is at least on string* s *produced by* $M(1^m)$, *such that* $A(s) = B(s)$, *for almost every lengths* m.

Suppose $\mathsf{NP} \overset{i.o.}{\neq} \mathsf{AM}$. Applying Theorem 19 with $M(1^m) = 1^m$ yields that for every $A \in \mathsf{BPP}$ and every $\epsilon > 0$, there exists $B \in \mathsf{DTIME}(2^{n^\epsilon})$ such that $A(1^m) = B(1^m)$, for almost every lengths m. Thus by Theorem 16 NP is $\mathsf{E}_{\alpha|\mathsf{oc}}$-meager for every $\alpha > 0$. □

8.2 BPP is easy or RP is small

The following result states that either RP is meager or ZPP equals BPP infinitely often.

Theorem 20 *Either* RP *is* $\mathsf{E}_{\alpha|\mathsf{oc}}$*-meager for every* $\alpha > 0$, *or* $\mathsf{ZPP} \overset{i.o.}{=} \mathsf{BPP}$.

Proof. The proof relies on Theorem 16 combined with the following result from [Kab00].

Theorem 21 *[Kab00] If* $\mathsf{ZPP} \overset{i.o.}{\neq} \mathsf{BPP}$, *then for every language* $A \in \mathsf{RP}$ *and every* $\epsilon > 0$, *there is a language* $B \in \mathsf{DTIME}(2^{n^\epsilon})$ *such that the following holds: For any length-preserving zero-error probabilistic polynomial-time Turing machine* M, *there is at least on string* s *produced by* $M(1^m)$ *such that* $A(s) = B(s)$, *for almost every lengths* m, *and* $M(1^m)$ *outputs a least one string with nonnegligeable probability.*

Suppose $\mathsf{ZPP} \overset{i.o.}{\neq} \mathsf{BPP}$. Applying Theorem 21 with $M(1^m) = 1^m$ satisfies the hypothesis of Theorem 16. Thus by Theorem 16 NP is $\mathsf{E}_{\alpha|\mathsf{oc}}$-meager for every $\alpha > 0$. □

9 Conclusion

The results in Section 8 should be compared to the zero-one laws obtained in Lutz's measure setting, see [Mel00] and [IM03] for more details. Although the results of Section 8 only hold for infinitely many input lengths, they require strategies which take less time to be computed compared to the martingales used in the measure setting, since the strategies can be computed in subexponential time whereas the martingales require exponential time to be computed. A further improvement would be to show that the results of section 8, hold in an almost-everywhere setting.

Acknowledgments
We'd like to thank E. Allender for pointing out a mistake in an earlier draft of this paper.

References

[AS94] E. Allender and M. Strauss. Measure on small complexity classes, with application for BPP. *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 807–818, 1994.

[AS95] K. Ambos-Spies. Resource-bounded genericity. *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 162–181, 1995.

[BDG90] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity II.* EATCS Monographs on Theorical Computer Science Volume 22, Springer Verlag, 1990.

[BDG95] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I.* EATCS Monographs on Theorical Computer Science Volume 11, Springer Verlag, 1995.

[Fen95] S. A. Fenner. Resource-bounded Baire category : a stronger approach. *Proceedings of the Tenth Annual IEEE Conference on Structure in Complexity Theory*, pages 182–192, 1995.

[IM03] R. Impagliazzo and P. Moser. A zero-one law for RP. *to be published in Computational Complexity Conference*, 2003.

[IW98] R. Impagliazzo and A. Widgerson. Randomness vs time de-randomization under a uniform assumption. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Scinece*, pages 734–743, 1998.

[Kab00] V. Kabanets. Easiness assumptions and hardness test: Trading time for zero error. *Proceedings of the Fifteenth Annual IEEE Conference on Computational Complexity*, pages 150–157, 2000.

[Lu00] C.-J. Lu. Derandomizing arthur-merlin games under uniform assumptions. *Proceedings of the Eleventh Annual International Symposium on Algorithms and Computation*, 2000.

[Lut90] J.H. Lutz. Category and measure in complexity classes. *SIAM Journal on Computing*, 19:1100–1131, 1990.

[Lut92] J.H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Science*, 44:220–258, 1992.

[Lut95] J.H. Lutz. Weakly hard problems. *SIAM Journal on Computing*, 24:1170–1189, 1995.

[Mel00] D. Melkebeek. The zero one law holds for BPP. *Theoretical Computer Science*, 244(1-2):283–288, 2000.

[Mos02] P. Moser. a generalization of Lutz's measure to probabilistic classes. *submitted*, 2002.

[Mos03] P. Moser. Baire's categories on small complexity classes. *to be published in Fundamentals of Computation Theory*, 2003.

[Pap94] C. Papadimitriou. *Computational complexity.* Addisson-Wesley, 1994.

[RS98] K. Regan and D. Sivakumar. Probabilistic martingales and BPTIME classes. *In Proc. 13th Annual IEEE Conference on Computational Complexity*, pages 186–200, 1998.

[Str97] M. Strauss. Measure on P- strength of the notion. *Inform. and Comp.*, 136:1:1–23, 1997.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'