Math. Log. Quart. **46** (2000) 4, 489 – 504

Mathematical Logic Quarterly

Looking for an Analogue of Rice's Theorem in Circuit Complexity Theory

Bernd Borchert and Frank Stephan[1)]

Mathematisches Institut, Universität Heidelberg,
Im Neuenheimer Feld 294, 69120 Heidelberg, Germany[2)]

Abstract. Rice's Theorem says that every nontrivial semantic property of programs is undecidable. In this spirit we show the following: Every nontrivial absolute (gap, relative) counting property of circuits is UP-hard with respect to polynomial-time Turing reductions. For generators [31] we show a perfect analogue of Rice's Theorem.

Mathematics Subject Classification: 03D15, 68Q15.

Keywords: Rice's Theorem, Counting problems, Promise classes, UP-hard, NP-hard, generators.

1 Introduction

One of the nicest theorems in Recursion Theory is Rice's Theorem [24, 25]. Informally speaking it says: Any nontrivial semantic property of programs is undecidable. More formally it can be stated this way:

Theorem 1.1 (Rice [24]). *Let A be any nonempty proper subset of the partial recursive functions. Then the halting problem or its complement is many-one reducible to the following problem: Given a program p, does it compute a function from A ?*

The theorem and its proof only use elementary notions of Recursion Theory. The most interesting point about Rice's Theorem is that it has messages to people in practical computing: It tells programmers that for example there is no program which finds infinite loops in given programs, or that there is no program which checks if some given program does a specified job.

Intrigued by the simple beauty of Rice's Theorem we tried to find some sister of it in Complexity Theory. We will present a concept based on Boolean circuits instead of programs, therefore our approach is different from the one of Kozen [19], Royer [20] and Case [10] who study problems on programs with given resource bounds.

[1)] The authors are grateful for discussions with Klaus Ambos-Spies, John Case, Wolfgang Merkle and André Nies. Furthermore the authors appreciate the comments by Lance Fortnow, Ulrich Hertrampf, Heribert Vollmer, Klaus Wagner and Gerd Wechsung after a talk at Dagstuhl. The second author is supported by Deutsche Forschungsgemeinschaft (DFG) grant Am 60/9-2. This work has been presented at the Kurt Gödel Colloquium 1997.

[2)] e-mail: {bb, fstephan}@math.uni-heidelberg.de

The main idea behind our approach is that for circuits versus Boolean functions there is a similar syntax/semantics dichotomy like there is for programs versus partial recursive functions: A circuit is the description of a Boolean function like a program is the description of a partial recursive function. In other words, circuits (programs) are the syntactical objects whereas Boolean functions (partial recursive functions) are the corresponding semantic objects. RICE's Theorem lives on this dichotomy of syntax and semantics: It says that semantic properties are impossible to recognize on the syntactical level. Similarly we look to what extent semantic properties of Boolean functions can be recognized from the syntactical structure of their circuits. Note that for example the question whether two syntactical objects describe the same semantic object is hard in both worlds: It is Π_2-complete for programs and co-NP-complete for circuits.

By these considerations, the perfect analogue of RICE's Theorem would be the following: Let A be any nonempty proper subset of the set of Boolean functions. Then the following problem is NP-hard: Given a circuit c, does it compute a function from A? An example for which this (generally wrong) claim is true is satisfiability: it is a semantic property because it does only depend on the Boolean function, not on the way the Boolean function is represented, and the satisfiability problem for circuits is NP-complete. But unfortunately, there is the following simple counterexample for the above claim. Let A be the set of Boolean functions which have the value 0 on the all-0-assignment. For a given circuit c this question can be computed in polynomial time. Therefore, the above claim stating NP-hardness of all nontrivial semantic properties of circuits is false (unless P=NP).

So we had to look for a more restrictive requirement for the set A for which a modification of the statement may be true. What we found is the (indeed very restrictive) requirement of *counting*: Let A be a set of Boolean functions for which membership in A only depends on the number of satisfying assignments, like for example the set of all Boolean functions which have at least one satisfying assignment, or the set of Boolean functions which have an odd number of satisfying assignments. We will call the corresponding sets of circuits *absolute counting problems*. In a similar fashion we will define *gap counting problems* and *relative counting problems*: they also incorporate the non-satisfying assignments in their definition. For example, the set of circuits which have more satisfying than non-satisfying assignments is a gap counting problem because it can be stated the following way: the gap (= difference) between the number of satisfying assignments and the number of non-satisfying assignments has to be greater than 0. And the same problem is a relative counting problem because it can be stated the following way: the relative number of satisfying assignments has to be greater than one half.

For each of these three types of counting problems a theorem in the fashion of RICE's Theorem can be shown, see Section 4, Conclusion 4.6.

> Any nontrivial absolute (gap, relative) counting property of circuits is UP-hard with respect to Turing reductions.

In Section 5, this result will be extended in terms of approximable sets and also in terms of randomized reductions.

In Section 6, we consider *generators* which were introduced by YAP [31] as a way of

representing Boolean functions. For generators a perfect analogue of RICE's Theorem can be found: Let A be any nonempty proper subset of the set of Boolean functions which not only depends on arity. Then the following problem is NP-hard: Given a generator g, does it describe a function from A?

HEMASPAANDRA and ROTHE [18] continue our line of research and study the absolute counting problem. Given any nontrivial set A, they show that the absolute counting problem for A is p-btt(1)-hard for $\mathrm{UP}_{O(1)}$. If A is furthermore P-constructibly infinite and coinfinite, then the absolute counting problem for A is p-m-hard for SPP.

2 Preliminaries

The standard notions of Theoretical Computer Science like words, languages, polynomial-time reductions, P, NP, etc., follow the book of PAPADIMITRIOU [22]. $A \oplus B$ is the join of two languages A and B, namely $A \oplus B = \{0x : x \in A\} \cup \{1x : x \in B\}$. Quite important for the present work is the notion of promise classes. These are classes where the number of accepting paths of a polynomial-time nondeterministic machine M to compute L satisfies the additional constraint that it does not take certain – theoretically possible – values. For example, if $L \in \mathrm{UP}$, then there is a machine M such that there is exactly one accepting paths for the case $x \in L$ and no accepting path for the case $x \notin L$. But it never occurs that there are 2 or more accepting paths. The following table gives an overview on promise classes and their conditions on the number $\#_1$ of accepting and $\#_0$ of rejecting paths.

Class	$x \notin L$	$x \in L$	not occurring
NP	$\#_1 = 0$	$\#_1 > 0$	–
PP	$\#_1 < \#_0$	$\#_1 \geq \#_0$	–
UP	$\#_1 = 0$	$\#_1 = 1$	$\#_1 > 1$
RP	$\#_1 = 0$	$\#_1 \geq \#_0$	$0 < \#_1 < \#_0$
FewP	$\#_1 = 0$	$0 < \#_1 < p(n)$	$\#_1 \geq p(n)$
HalfP	$\#_1 = 0$	$\#_1 = \#_0$	$\#_1 \neq 0 \wedge \#_1 \neq \#_0$
SPP	$\#_1 = \#_0$	$\#_1 = \#_0 + 2$	$\#_1 \neq \#_0 \wedge \#_1 \neq \#_0 + 2$
BPP	$\#_1 \leq \varepsilon \cdot \#_0$	$\#_0 \leq \varepsilon \cdot \#_1$	$\#_1 > \varepsilon \cdot \#_0 \wedge \#_0 > \varepsilon \cdot \#_1$

In this table, ε is a fixed constant with $0 < \varepsilon < 1$ and $p(n)$ is a value polynomial in the length n of the input. BERTHIAUME and BRASSARD [5] introduced the class HalfP using the name $\mathrm{C}_{=}\mathrm{P}$[half].

A central notion of this paper is the notion of a *Boolean function* which is defined to be a mapping from $\{0,1\}^n$ to $\{0,1\}$ for some integer $n \geq 0$ which is called its *arity*. *Circuits* are a standard way of representing Boolean functions. We will just assume that they are encoded as words in some standard way, for the details concerning circuits we refer for example to [22]. Remember that a given circuit can be evaluated on a given assignment in polynomial time [20]. Each circuit c describes a Boolean function $F(x_1, \ldots, x_n)$. Note that here we have an example of the classical syntax/semantics dichotomy like we have it for programs: The circuits (programs) are the syntactical objects, which we have as finite words at our fingertips, whereas

the Boolean functions (partial recursive functions) are the corresponding semantic objects far away in the mathematical sky.[3)]. Note that every Boolean function (partial recursive function) can be represented by a circuit (program), in fact it is represented by infinitely many circuits (programs). The problem whether two circuits (programs) represent the same Boolean function (partial recursive function) is co-NP-complete (Π_2-complete).

As it was pointed out in the introduction one would like to have a theorem like: Every nontrivial semantic property of circuits is NP-hard. Unfortunately, there are counterexamples for this statement unless P=NP. First of all there is the counterexample consisting of the set of circuits with an odd number of satisfying assignments, this set is complete for $\bigoplus$P and not known to be NP-hard or co-NP-hard. So we have to replace in the statement above the class NP by some class which is contained in NP and $\bigoplus$P. The class UP would be a natural choice for that. But unfortunately there is an even stronger counterexample: the set of all circuits which evaluate to 0 on the all-0-assignment. This property of circuits is in fact a semantic property and it is nontrivial, but it can be checked with a polynomial-time algorithm. So we can only hope for an analogue of Rice's Theorem (stating UP-hardness) if we restrict ourselves to stronger semantic properties. We tried several approaches, for example by considering the equivalence relations like Boolean isomorphism presented in [1, 9]. Note that the above counterexample is also a counterexample under Boolean isomorphism. The restriction for which we could find the intended hardness result is the restriction to the counting properties which will be introduced in the next section.

3 Three types of counting problems on circuits

Given a Boolean function one can ask different question about the number of satisfying assignments: (a) what is the number of satisfying assignments? (b) what is the difference between the number of satisfying and non-satisfying assignments? (c) what is the share of the satisfying assignments compared with the total number of assignments? For a fixed arity, these three approaches are equivalent, but as we deal with Boolean functions of all arities, the approaches give three different classes of counting problems.

More formally, let, for a circuit c, $\#_0(c)$ and $\#_1(c)$ denote the number of non-satisfying assignments and satisfying assignments, respectively, of the Boolean function represented by c. According to the three approaches (a), (b) and (c) from above we will introduce the following three types of counting problems. Note that the answer to (a) is a natural number, the answer to (b) an integer and the answer to (c) a dyadic number in the interval $[0, 1]$, we will denote this set by

$$\mathbb{D} = \{\frac{m}{2^n} : n,\, m \in \mathbb{N},\, 0 \leq m \leq 2^n\}.$$

[3)] One might argue that here we have a different situation than in the case of partial recursive functions because Boolean functions are finite objects: one can represent an n-ary Boolean function just by the length-2^n $\{0,1\}$-valued sequence of its values on the 2^n assignments. This in fact guarantees decidability of the usual semantic questions, but we are interested in finer questions: even if semantic properties are decidable, how difficult is it to decide them? And the difficulty for deciding the semantic questions (like satisfiability) stems basically from the fact that Boolean functions are not given as the length-2^n 0-1-sequence of the function values but in a *compressed* way, namely as circuits.

Definition 3.1.

(a) Let A be a subset of $\mathbb{N}$. The *absolute counting problem for* A, Abs-Count(A), is the set of all circuits c such that $\#_1(c) \in A$.

(b) Let A be a subset of $\mathbb{Z}$. The *gap counting problem for* A, Gap-Count(A), is the set of all circuits c such that $\#_1(c) - \#_0(c) \in A$.

(c) Let A be a subset of $\mathbb{D}$. The *relative counting problem for* A, Rel-Count(A), is the set of all circuits c such that the relative number of accepting assignments is in A:

$$c \in \text{Rel-Count}(A) \Leftrightarrow \frac{\#_1(c)}{\#_0(c) + \#_1(c)} \in A.$$

Examples of absolute counting problems. The satisfiability problem for circuits, which we will denote here as SAT (though SAT traditionally refers to the satisfiability problem on CNF's), is by its definition an absolute counting problem, that is, SAT = Abs-Count($\{1, 2, 3, \ldots\}$). Remember that SAT is NP-complete. Likewise, the set of unsatisfiable circuits equals the set Abs-Count($\{0\}$), this problem is co-NP-complete. Another example is the set of circuits with an odd number of satisfying assignments, by definition it equals Abs-Count($\{1, 3, 5, \ldots\}$), this problem is $\bigoplus$P-complete. Another example is the set 1-SAT consisting of the circuits with exactly one satisfying assignment, that is, 1-SAT = Abs-Count($\{1\}$), the complexity class for which this problem is complete is usually called US, see [6], or also 1-NP.

Examples of gap counting problems. The set C$_=$SAT of circuits which have as many satisfying as non-satisfying assignments, is a gap counting problem: C$_=$SAT = Gap-Count($\{0\}$). The set PSAT of circuits which have at least as many accepting as non-accepting assignments, is a gap counting problem: PSAT = Gap-Count($\{0, 1, 2, 3, \ldots\}$). Remember that C$_=$SAT and PSAT are complete for the classes C$_=$P and PP, respectively. Another example is the set Gap-2-SAT consisting of the circuits with exactly two more satisfying than non-satisfying assignments, that is, Gap-2-SAT = Gap-Count($\{2\}$).

Examples of relative counting problems. The two gap counting problems PSAT and C$_=$SAT from above are also relative counting problems:

$$\text{C}_=\text{SAT} = \text{Rel-Count}(\{\tfrac{1}{2}\}) \text{ and } \text{PSAT} = \text{Rel-Count}(\{x \in \mathbb{D} : x \geq \tfrac{1}{2}\}).$$

SAT = Rel-Count($\mathbb{D} - \{0\}$) is also a relative counting problem. The tautology problem equals Rel-Count($\{1\}$).

Note that only relative counting has the following natural property: If we have a circuit $c(x_1, \ldots, x_m)$ and add some dummy variables $x_{m+1}, \ldots, x_n$ so that the whole new circuit $c'(x_1, \ldots, x_m, \ldots, x_n)$ represents a Boolean function on $n \geq m$ inputs, then we have that $c(x_1, \ldots, x_m) \in$ Rel-Count(A) iff $c'(x_1, \ldots, x_m, \ldots, x_n) \in$ Rel-Count(A). In a way one would consider the Boolean functions represented by c and c' to be "basically the same". So if we identify Boolean functions modulo independent variables, then only relative counting respects this natural identification.

A *counting problem* in a general sense we define in the following way: Let a sequence (A_n) be given for which A_n is a subset of $\{0, \ldots, 2^n\}$. The *counting problem for* (A_n) is the set of all circuits $c(x_1, \ldots, x_n)$ such that $\#_1(c) \in A_n$, see [15] for an analogous definition of (general) counting classes. In this way, absolute, gap and relative counting problems are counting problems. It is easy to give an

example of a (general) counting problem which is nontrivial but in P, for example the set of all circuits with an odd arity (it is the counting class for the sequence $\emptyset, \{0,1,2\}, \emptyset, \{0,1,\ldots,8\}, \emptyset, \{0,1,\ldots,32\}, \emptyset, \ldots$).

It was already mentioned that the three types of counting problems from Definition 3.1 are incomparable as mathematical sets. But the question appears if they are comparable in terms of p-m-complexity. For example, the tautology problem, which is a relative counting problem, is provably not an absolute counting problem, nevertheless there is an absolute counting problem, namely the non-satisfiability problem, which has the same p-m-complexity (both are co-NP-complete). But even for this weaker form of comparison the three types of counting problems seem to be incomparable: PSAT and C$_=$SAT are gap and relative counting problems but do not seem to be p-m-equivalent to some absolute counting problem. SAT is an absolute and relative counting problem but does not seem to be p-m-equivalent to some gap counting problem. 1-SAT is an absolute counting problem but does not seem to be p-m-equivalent to some gap or relative counting problem. Gap-2-SAT is a gap counting problem but does not seem to be p-m-equivalent to some absolute or relative counting problem.

R e m a r k . Analogously to the way we defined the three types of counting problems we can define three types of counting classes: For a given subset A of $\mathbb{N}$ ($\mathbb{Z}$, $\mathbb{D}$) the *absolute* (*gap, relative*) *counting class for* A consists of the languages L for which there is a polynomial-time nondeterministic machine M such that a word x is in L iff the number of accepting paths (the difference of the number of accepting paths and non-accepting paths, the share of the accepting paths compared with the total number of paths) of M on input x is in A. This definition of *gap countable classes* equals the definition of *nice gap definable classes* by Fenner, Fortnow and Kurtz [13]. As a special case of the main result in [8, 30] it follows that an absolute (gap, relative) counting problem is p-m-complete for the corresponding absolute (gap, relative) counting class. In other words, absolute (gap, relative) counting problems and the corresponding absolute (gap, relative) counting classes are just two sides of the same medal. Note that the p-m-comparability question we discussed above is therefore equivalent to the set comparability question of the three types of counting classes.

4 Some Rice-style theorems for counting problems

In this section we will state and prove the theorems which show UP-hardness of all nontrivial counting problems of the three types defined in the previous chapter. Remember that the class UP, which was defined first in [28], is the promise class consisting of the languages L such that there is a polynomial-time nondeterministic machine M such that on every input the machine M has at most one accepting path and an input x is in L if M running on input x has an accepting path. Such machines are called *unambiguous*. By definition UP is a subset of NP. A classical result tells that UP equals P if and only if one-way functions do not exist (see [14]). The primality problem is a typical example of a problem in UP not known to be in P (see [12]).

Below we use the notion of a class $\mathcal{C}$ being reducible to a language L, this is just a short way of saying that every language in the class $\mathcal{C}$ is reducible to L. The following theorem implies that any nontrivial absolute counting property of circuits is UP-hard.

T h e o r e m 4.1. *Let A be any nonempty proper subset of $\mathbb{N}$. Then at least one of the following three classes is p-m-reducible to* Abs-Count(A)*:* NP, co-NP, UP$\oplus$ co-UP.

P r o o f. The proof distinguishes three cases.

C a s e 1. A has a maximum a. Then co-NP is p-m-reducible to Abs-Count(A): Let a language L in co-NP be given and let M be a machine for L in the sense that $x \in L$ iff no path of $\mathsf{M}(x)$ is accepting. Let $m(x)$ denote the number of accepting paths of $\mathsf{M}(x)$. Cook [11] (see also Ladner [20]) established a method to construct in polynomial time a circuit $\mathrm{Cook}_x^{\mathsf{M}}$ with inputs $y_1, \ldots, y_n$ (n depends on x and is bounded by a polynomial in the length of x) such that accepting computation paths of $\mathsf{M}(x)$ and satisfying assignments of $\mathrm{Cook}_x^{\mathsf{M}}(y_1, \ldots, y_n)$ correspond to each other. Therefore, $\mathrm{Cook}_x^{\mathsf{M}}(y_1, \ldots, y_n)$ evaluates to 1 for exactly $m(x)$ assignments. Using a additional variables $z_1, \ldots, z_a$ the circuit given by the following specification evaluates exactly $a + m(x)$ assignments to 1:

$$d_{x,a}^{\mathsf{M}}(y_1, \ldots, y_n, z_1, \ldots, z_a) = \begin{cases} \mathrm{Cook}_x^{\mathsf{M}}(y_1, \ldots, y_n) & \text{if } z_1 + \cdots + z_a = 0, \\ 1 & \text{if } z_1 + \cdots + z_a = 1 \\ & \quad \text{and } y_1 + \cdots + y_n = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Thus $d_{x,a}^{\mathsf{M}}$ is in Abs-Count(A) iff $m(x) = 0$ iff $x \in L$. So the mapping $x \longmapsto d_{x,a}^{\mathsf{M}}$ gives a p-m-reduction from L to Abs-Count(A).

C a s e 2. $\overline{A}$ has a maximum b. Given a set L in NP recognized by the nondeterministic machine M, an analogous construction as in the previous case is used in order to obtain a circuit $d_{x,b}^{\mathsf{M}}$ which evaluates exactly $b + m(x)$ assignments to 1. Now $x \in L$ iff $m(x) > 0$ iff $d_{x,b}^{\mathsf{M}} \in$ Abs-Count(A). And so one obtains the desired p-m-reduction.

C a s e 3. Neither A nor $\overline{A}$ has a maximum. Then there are $a \in A$ with $a + 1 \notin A$ and $b \notin A$ with $b + 1 \in A$. For a language $L = 0L' \cup 1L''$ in UP $\oplus$ co-UP let M' and M'' be the unambiguous machines for L' and L'', respectively. The mapping which assigns to an input $0x$ the circuit $d_{x,a}^{\mathsf{M}'}$ and to an input $1x$ the circuit $d_{x,b}^{\mathsf{M}''}$ realizes the p-m-reduction from L to Abs-Count(A). □

Hemaspaandra and Rothe [18] improved this result. Let Const denote the class of all sets which are p-btt-reducible to some set in UP.

T h e o r e m 4.2 (Hemaspaandra and Rothe [18]). *Let A be a nontrivial subset of $\mathbb{N}$. Then at least one of the three classes* NP, co-NP, Const *is p-m-reducible to* Abs-Count(A).

Furthermore, they introduce the notion of P-constructibly bi-infinite sets and use it to obtain the following result for the relation to the class SPP (defined below).

T h e o r e m 4.3 (Hemaspaandra and Rothe [18]). *If A is finite or cofinite, then* Abs-Count(A) *is in the Boolean Hierarchy over* NP *and thus there is a relativized world where* SPP *is not p-m-reducible to* Abs-Count(A). *Furthermore, if A is* P*-constructibly bi-infinite, then* SPP $\leq_{\mathrm{m}}^{\mathrm{p}}$ Abs-Count(A).

Gupta [16] and Ogiwara and Hemachandra [21] introduced the class SPP as the promise class consisting of the languages L such that there is a polynomial-time nondeterministic machine M such that for every input x for the machine M either half of the computation paths are accepting or half of them plus 1 are accepting, and

$x \in L$ if the second case is true. Note that the class SPP contains both UP and co-UP. Fenner, Fortnow and Kurtz [13] proved the following result which states that the class SPP is p-m-reducible to any gap counting problem.

Theorem 4.4. (Fenner, Fortnow and Kurtz [13]). *Let A be any nonempty proper subset of $\mathbb{Z}$. Then* SPP $\leq^{\mathrm{p}}_{\mathrm{m}}$ Gap-Count(A).

Now to obtain UP-hardness for absolute and gap counting problems we turn to relative counting problems and will show UP-hardness for the more powerful polynomial-time Turing reducibility. Note that the classes NP, co-NP and SPP contain UP.

Theorem 4.5. *Let A be any nonempty proper subset of $\mathbb{D}$. Then* Rel-Count(A) *is p-m-complete for* NP *or* co-NP*, or* SPP $\leq^{\mathrm{p}}_{\mathrm{T}}$ Rel-Count(A).

Proof. For the ease of notation, let $A(p)$ be 1 for $p \in A$ and 0 for $p \notin A$. First consider the special case that $A(p) = A(q)$ for all dyadic numbers p, q with $0 < p < q < 1$. In this special case, A is one of the following six sets: $\{0\}$, $\{1\}$, $\{0,1\}$, $\mathbb{D} - \{0\}$, $\mathbb{D} - \{1\}$, $\mathbb{D} - \{0,1\}$. It is easy to see that the relative counting problems for first three sets are co-NP-complete, for example, Rel-Count($\{1\}$) is the tautology problem. Similarly the relative counting problems for the last three sets are NP-complete, for example, Rel-Count($\mathbb{D} - \{0\}$) is the satisfiability problem. So it remains the case where there are dyadic numbers p, q such that $0 < p < q < 1$ and $A(p) \neq A(q)$. It will be shown that in this case SPP can be polynomial-time Turing reduced to Rel-Count(A). Let M be a machine which witnesses that a language L is in SPP. Consider for an input x the circuit $\mathrm{Cook}^{\mathsf{M}}_x(y_1, \ldots, y_n)$ defined in the proof of Theorem 4.1. Recall that if $x \in L$, then $\mathrm{Cook}^{\mathsf{M}}_x$ evaluates $2^{n-1}+1$ assignments to 1, and if $x \notin L$, then $\mathrm{Cook}^{\mathsf{M}}_x$ evaluates 2^{n-1} assignments to 1. Furthermore there is an m such that p and q are multiples of 2^{-m} and thus also of 2^{-n-m} for all n. Now the Turing-reduction works as follows:

(i) Search for multiples p', q' of 2^{-m-n} with $p \leq p' < q' \leq q$, $q' - p' = 2^{-m-n}$ and $A(p') \neq A(q')$ by interval search.

A query to A can be translated into a query to Rel-Count(A) as follows: Let $r = 0.a_1 \ldots a_k < 1$ be a dyadic number and let c_r denote the circuit which assigns to $(y_1, \ldots, y_k)$ the value 1 iff $0.y_1 \ldots y_k < 0.a_0 \ldots a_k$. r is in A iff $c_r \in$ Rel-Count(A). Using this mechanism it is possible to find p' and q' with interval search: Starting with $p = p'$ and $q = q'$ one takes that $m+n$-bit dyadic number r which is nearest to $(p'+q')/2$ and finds out whether $A(p') = A(r)$ or $A(q') = A(r)$. In the first case, p' is replaced by r, in the second, q' is replaced by r. This search is continued until the difference between p' and q' is 2^{-m-n}. Note that $A(p) = A(p') \neq A(q) = A(q')$.

(ii) Now a circuit d_x is computed with Rel-Count(A)(d_x) $= A(q')$ for $x \in L$ and Rel-Count(A)(d_x) $= A(p')$ for $x \notin L$. Let $z = 0.z_1 \ldots z_{m+n}$ be the dyadic number determined via the binary representation of the variables. Then we set

$$d_x(z_1, \ldots, z_{m+n}) = \begin{cases} \mathrm{Cook}^{\mathsf{M}}_x(z_{m+1}, \ldots, z_{m+n}) & \text{if } z < 2^{-m}, \\ 1 & \text{if } 2^{-m} \leq z < p' + 2^{-m-1}, \\ 0 & \text{if } p' + 2^{-m-1} \leq z. \end{cases}$$

So the relative number of accepting assignments is the sum of the $p' - 2^{-m-1}$ hard

wired assignments from the second line of the case-distinction plus 2^{-m-1} (resp. $2^{-m-1}+2^{-m-n}$) from the circuit c_x in the case of $x \notin L$ (resp. $x \in L$). Then the whole relative number is p' for $x \notin L$ and q' for $x \in L$. It follows that $x \in L$ iff Rel-Count$(A)(d_x) = A(q')$. So the last query whether d_x is in Rel-Count(A) completes the decision procedure for L.

In short words: Part (i) of the construction uses the fact that $A \leq^{\mathrm{p}}_{\mathrm{m}}$ Rel-Count(A) in order to search sufficiently close dyadic numbers p' and q' between p and q such that $A(p') \neq A(q)$, and step (ii) produces a circuit d_x whose relative number of satisfying assignments is p' for $x \notin L$ and q' for $x \in L$. So $L(x)$ can be computed with $m+n+1$ queries to Rel-Count(A). Therefore, in this second case, Rel-Count(A) is SPP-hard with respect to polynomial-time Turing reductions. □

The preceding three Theorems 4.1, 4.4, 4.5 can be summarized by the following conclusion:

Conclusion 4.6. *Any nontrivial absolute* (*gap, relative*) *counting property of circuits is* UP*-hard with respect to polynomial-time Turing reducibility. In particular, a nontrivial absolute* (*gap, relative*) *counting problem on circuits is not solvable in polynomial-time unless* P = UP.

Berthiaume and Brassard [5] considered the following class HalfP consisting of all sets L which have a nondeterministic machine M such that $x \in L$ iff exactly the half of the total number of paths are accepting, and $x \notin L$ iff no path is accepting. The class HalfP lies between P and the Quantum Computation Class QP (see [5]). The next result shows that either HalfP or co-HalfP is a lower bound for any non-trivial relative counting problem.

Theorem 4.7. *If* Rel-Count(A) *is not trivial, then* HalfP $\leq^{\mathrm{p}}_{\mathrm{m}}$ Rel-Count(A) *or* co-HalfP $\leq^{\mathrm{p}}_{\mathrm{m}}$ Rel-Count(A).

Proof. Since Rel-Count(A) is not trivial, there are numbers $p, q \in \mathbb{D}$ such that $A(p) \neq A(q)$. Since $A(1/2)$ must be equal to either $A(p)$ or $A(q)$ but not to both, one can replace one of the numbers p and q by $1/2$. Without loss of generality $p < 1/2$ and $q = 1/2$. Let m be the number of digits in the dual representation of $0.p_1p_2\ldots p_m$ of p if $p \neq 0$, and $m = 1$ and $p_1 = 0$ if $p = 0$. Let L be a language in HalfP and M a machine witnessing this. Given x, let n be the arity of the circuit $\mathrm{Cook}^{\mathsf{M}}_x$. Furthermore, let $z = 0.z_1\ldots z_m$ be the dyadic number determined by the binary representation of the first m variables and let

$$d_x(z_1,\ldots,z_{m+n}) = \begin{cases} \mathrm{Cook}^{\mathsf{M}}_x(z_{m+1},\ldots,z_{m+n}) & \text{if } p \leq z < 1-p, \\ 1 & \text{if } z < p, \\ 0 & \text{if } 1-p \leq z. \end{cases}$$

The share of accepting assignments is $p + 2\cdot(1-p)\cdot 2^{-m}\cdot a$, where a is the share of accepting assignments of the circuit $\mathrm{Cook}^{\mathsf{M}}_x$. One can compute that the share of accepting assignments of d_x is p for $a = 0$ and $q = 1/2$ for $a = 1/2$. So the share is p for $x \notin L$ and q for $x \in L$. Thus the mapping $x \longmapsto d_x$ is a p-m-reduction from L to Rel-Count(A) in the case $A(1/2) = 1$ and a p-m-reduction from $\overline{L}$ to Rel-Count(A) in the case $A(1/2) = 0$. So HalfP is p-m-reducible to Rel-Count(A) in the case $1/2 \in A$ and co-HalfP is p-m-reducible to Rel-Count(A) otherwise. □

5 Extensions and limitations of the main results

In this section first the result is extended to randomized reductions and computations. After that it is shown that no nontrivial absolute (gap, relative) counting problem on circuits is approximable unless P = UP. Furthermore it is pointed out that it is unlikely that Theorem 4.5 holds with polynomial-time many-one-reduction in place of the polynomial-time Turing reduction.

5.1 Randomized computations

Valiant and Vazirani [29] showed, that using randomized reductions, detecting unique solutions is as hard as solving the satisfiability problem. In particular they showed that every algorithm f which satisfies the following specification also already allows to solve the satisfiability problem in a randomized context: If the circuit x has a solutions and $a \leq 1$, then $f(x) = a$. The algorithms to reduce UP to the counting problems in Theorems 4.1, 4.4 and 4.5 satisfy these requirements. So they allow to decide the set SAT $= \{x : x$ is a circuit and x has a solution$\}$ via a nondeterministic machine which has no accepting path for $x \notin$ SAT and which has more accepting than rejecting paths for $x \in$ SAT. Thus the following holds for all three counting problems, in particular for relative counting:

Theorem 5.1. *If* $B =$ Rel-Count(A) *is not trivial, then* NP $\subseteq$ RP^B.

For absolute and gap counting, the result can be improved by showing that SAT is randomized polynomial-time reducible to B or to $\overline{B}$. Valiant and Vazirani [29] defined that a randomized (many-one) polynomial-time reduction from some set A to another set B is given via a machine M which computes for every x and path p a circuit $M(x, p)$ such that

$$x \in A \Rightarrow M(x,p) \in B \text{ for at least } n(x)/q(\text{length of } x) \text{ paths } p,$$
$$x \notin A \Rightarrow M(x,p) \notin B \text{ for all paths } p,$$

where q is a suitable polynomial and $n(x)$ is the number of computation paths of M. Valiant and Vazirani [29, Theorem 1.1] constructed a randomized polynomial-time reduction from SAT to USAT$_Q$, where

$$\text{USAT}_Q(x) = \begin{cases} a & \text{if } x \text{ has } a \text{ solutions and } a \leq 1, \\ Q(x) & \text{if } x \text{ has at least two solutions,} \end{cases}$$

for some $\{0,1\}$-valued function Q and where the reduction is independent of the values $Q(x)$ for all x. The result is a direct combination of this construction and the constructions for p-m-reducing UP to B or $\overline{B}$ in Theorems 4.1 and 4.4 which indeed are p-m-reductions of some suitable set USAT$_Q$ to B or $\overline{B}$, respectively.

Theorem 5.2. SAT *or its complement is randomized polynomial-time reducible to any nontrivial absolute and gap counting problem.*

5.2 Approximable sets

A set A is approximable (see [4]) iff there is a constant j and an algorithm which computes for each input $x_1, \ldots, x_j$ in polynomial time j bits $y_1, \ldots, y_j$ such that $A(x_h) = y_h$ for some h. Beigel [2, 3] analyzed the notion of approximable sets and

showed that no NP-hard set is approximable unless P = UP. This result can be transferred to the following theorem which is an extension of Conclusion 4.6.

Theorem 5.3. *If* P $\neq$ UP*, then no nontrivial absolute* (*gap, relative*) *counting problem is approximable.*

Proof. The proofs are all direct combinations of Beigel's techniques with those to show the UP-hardness of these sets. Thus we restrict ourselves to show the most involved case of relative counting sets.

If a problem is NP-complete or co-NP-complete, then it is not approximable under the hypothesis P $\neq$ UP (see [2, 3]). So one has only to adapt the main case in the proof of Theorem 4.5.

So let L be any language in UP, note that $UP \subseteq$ SPP. The first part of the Turing reduction from L to Rel-Count(A) is exactly the same as in the proof of Theorem 4.5. In the second part it starts to differ at the definition of the circuit d_x. Based on the definition of d_x, n variants $d_{n,i}$ are defined via fixing one variable to 1 in the circuit $\mathrm{Cook}_x^{\mathsf{M}}$:

$$d_{x,i}(z_1,\ldots,z_{m+n}) = \begin{cases} \mathrm{Cook}_x^{\mathsf{M}}(z_{m+1},\ldots,z_{m+n}) & \text{if } z < 2^{-m} \text{ and } z_{m+i} = 1, \\ 0 & \text{if } z < 2^{-m} \text{ and } z_{m+i} = 0 \\ & \quad \text{or } p' + 2^{-m-1} \leq z, \\ 1 & \text{if } 2^{-m} \leq z < p' + 2^{-m-1}. \end{cases}$$

Recall that for $x \in L$ there is exactly one satisfying assignment $(a_1,\ldots,a_n)$ and for $x \notin L$ no one. So the relative number of the accepted assignments of $d_{x,i}$ is $q' = p' + 2^{-m-n}$ if $x \in L$ and $a_i = 1$, and p' otherwise. As Beigel [3, Theorem 9] pointed out, there is an algorithm which computes for input $(d_{x,1},\ldots,d_{x,n})$ in polynomial time $O(n^j)$ n-bit-vectors v such that one of these vectors is the characteristic function of A on the input-vector. So for each such $v = (v_1,\ldots,v_n)$ one computes the assignment $(a_1,\ldots,a_n)$ given via $a_i = 1$ for $v_i = A(q')$ and $a_i = 0$ for $v_i = A(p')$. If $\mathrm{Cook}_x^{\mathsf{M}}$ has a satisfying assignment, then one of these $(a_1,\ldots,a_n)$ must be one. By evaluating $\mathrm{Cook}_x^{\mathsf{M}}$ on these assignments it is found out whether $x \in L$ or $x \notin L$. Thus if Rel-Count(A) is approximable, then L is in P and P = UP. □

5.3 Relative counting and BPP

Hartmanis and Hemachandra [17, Theorem 4.1] show that in some recursively relativized world the class BPP has no complete set, that is, there is no set A with BPP $= \{L : L \leq_{\mathrm{m}}^{\mathrm{p}} A\}$. On the other hand, Theorem 5.4 below shows that it is still possible to find a set A such that BPP $= \{L : L$ is recursive and $L \leq_{\mathrm{m}}^{\mathrm{p}} A\}$. This A can – at least in the relativized worlds considered by Hartmanis and Hemachandra [17] – not be recursive. So an interesting question is what kind of properties such a set A still can satisfy. Within the context of the present work, it is quite natural to ask whether such a set can have the form of some nontrivial counting problem, and Theorem 5.4 gives an affirmative answer. A corollary from this result is that there are relative counting-problems which are not only p-m-hard for UP but also p-m-hard for BPP.

Recall that BPP is the class of all languages L such that there is an $\varepsilon > 0$ and a polynomial-time nondeterministic machine M such that for all inputs x the share of

accepting paths (all of them must have the same length) is either less than $\frac{1}{2} - \varepsilon$ or greater than $\frac{1}{2} + \varepsilon$; x is accepted in the second case.

Theorem 5.4. *There is a set $A \subseteq \mathbb{D}$ such that for every recursive set L,*

$$L \leq^{\mathrm{p}}_{\mathrm{m}} \text{Rel-Count}(A) \quad \textit{iff} \quad L \in \text{BPP}.$$

Proof. There are uncountably many reals r between 0 and 1 and for each such real r, the set $A_r = \{q \in \mathbb{D} : q < r\}$ is different from all other sets $A_{r'}$ with $r' \neq r$. Thus, using a natural representation of dyadic numbers by words, there is a set A_r of this form which is not recursively enumerable. So fix such this real r and the set $A = A_r$.

($\Rightarrow$). Let L be a recursive set which is p-m-reducible to Rel-Count(A) via a function f. Let $m(x)$ denote for each x the relative number of satisfying assignments of the circuit $f(x)$, $m(x)$ can be computed from $f(x)$ using exponential time. Since $m(x)$ is in A iff x is in L, the set $B = \{q \in \mathbb{D} : (\exists x \in L)\,[q \leq m(x)]\}$ is a recursively enumerable subset of A. Since A is not recursively enumerable but B is, B is different from A and the supremum of B must be below that of A: $\sup(B) < r$. There is a dyadic number s strictly between these two numbers. So there is some $\varepsilon > 0$ such that $\sup(B) < s - 2\varepsilon$ and $r > s + 2\varepsilon$. Let $n(x)$ be the number of inputs of circuit $f(x)$. Now the following machine M witnesses that L is in BPP.

$$\mathsf{M}(x)(y_0, \ldots, y_{n(x)}) = \begin{cases} 1 & \text{if } y_0 = 0 \text{ and } f(x) \text{ evaluates } (y_1, \ldots, y_{n(x)}) \text{ to } 1 \\ & \quad \text{or } y_0 = 1 \text{ and } 0.y_1 \ldots y_{n(x)} > s, \\ 0 & \text{otherwise.} \end{cases}$$

The relative number $k(x)$ of the accepting assignments is the sum of two numbers, the number $m(x)/2$ from the simulation of $f(x)$ and the number $(1 - s)/2$ from the hard-wired paths with $y_0 = 1$. So if $x \in L$, then $m(x) \in A$, $m(x) \leq s - 2\varepsilon$ and $k(x) \leq \frac{1}{2} - \varepsilon$, and if $x \notin L$, then $m(x) \notin A$, $m(x) \geq s + 2\varepsilon$ and $k(x) \geq \frac{1}{2} + \varepsilon$. Since the constant ε does not depend on x and $n(x)$, M witnesses that L is in BPP.

($\Leftarrow$). Now let $L \in$ BPP. There is a real number ε with $|r - \frac{1}{2}| < \varepsilon < \frac{1}{2}$. For L there is now a BPP-machine M which works with this ε, see e. g. [22, Chapter 11]. Let M have path length $n(x)$ on input x. Now one assigns to each x the circuit $\neg\text{Cook}^{\mathsf{M}}_x(y_1, \ldots, y_{n(x)})$. The relative number of the satisfying assignments for each circuit $\text{Cook}^{\mathsf{M}}_x$ is above $\frac{1}{2} + \varepsilon > r$ for $x \notin L$ and below $\frac{1}{2} - \varepsilon < r$ for $x \in L$. So the mapping $x \longmapsto \neg\text{Cook}^{\mathsf{M}}_x$ is a p-m-reduction from L to Rel-Count(A). □

Remark. Book, Lutz and Wagner [7] showed some analogous result for polynomial-time Turing reducibility and random sets in place of p-m-reducibility and counting problems: There is some Martin-Löf random set A such that a recursive set L is in BPP iff $L \leq^{\mathrm{p}}_{\mathrm{T}} A$.

For the classes NP, PP, $\mathrm{C}_{=}\mathrm{P}$ and $\mathrm{Mod}_k\mathrm{P}$ a better result is possible since each of these classes is equal to the class of all languages which are p-m-reducible to some suitable relative counting problem. For example, $\mathrm{C}_{=}\mathrm{P}$ is the set of languages p-m-reducible to Rel-Count($\{1/2\}$) and NP is the set of languages p-m-reducible to Rel-Count($\{q : q > 0\}$).

The proof of Theorem 5.4 is relativizable. The next result shows that the corresponding result for UP depends on the world chosen. It holds of course for those worlds

where UP = BPP. But one can construct a relativized world where it fails as follows: Theorem 4.7 shows that either HalfP $\leq_{\mathrm{m}}^{\mathrm{p}}$ Rel-Count(A) or co-HalfP $\leq_{\mathrm{m}}^{\mathrm{p}}$ Rel-Count(A) for any $A \subseteq \mathbb{D}$. The theorem that P = UP $\subset$ NP for some relativized world (see [23]) can be modified such that P = UP $\subset$ HalfP (and therefore also UP $\subset$ co-HalfP). In this relativized world, there is, for any non-trivial $A \subseteq \mathbb{D}$, a recursive set L outside UP with $L \leq_{\mathrm{m}}^{\mathrm{p}}$ Rel-Count(A). Therefore Theorem 4.7 does not have a parallel for UP in this relativized world.

Corollary 5.5. *There is a relativized world such that there is no $A \subseteq \mathbb{D}$ with* UP $= \{L : L$ *is recursive and* $L \leq_{\mathrm{m}}^{\mathrm{p}}$ Rel-Count(A)$\}$.

6 A perfect analogue of Rice's theorem for generators

It was mentioned before that the problem whether two programs compute the same partial recursive function is complete for Π_2 in the Arithmetical Hierarchy, whereas the problem whether two circuits compute the same Boolean function is complete for co-NP $= \Pi_1^{\mathrm{p}}$ in the Polynomial-Time Hierarchy. So the difficulty of the two equivalence problems is on different levels in the respective hierarchies and maybe that is the reason why a perfect analogue of Rice's Theorem does not work for circuits. Therefore, we have been looking for a representation of Boolean functions such that the equivalence problem is complete for Π_2^{p}. Such a representation is given by *generators* which are defined the following way (the first definition by Yap [31] did not have the indicator bit i which was added by Schöning [27] in order to be able to represent, for each n, the n-ary constant 0-function).

Definition 6.1 (Yap [31]). A *generator* is a Boolean circuit, with k inputs and $n+1$ outputs $i, r_1, \ldots, r_n$. The function computed by it is the n-ary Boolean function which has value 1 for an input $(a_1, \ldots, a_n)$ iff there is a k-ary assignment for the circuit, such that the output i has the value 1 and the output vector $(r_1, \ldots r_n)$ has the value $(a_1, \ldots, a_n)$.

In other words, ignoring the indicator bit, the function computed by a generator is basically the characteristic function of the range of the function described by the circuit. Generators are the nonuniform counterpart of polynomial-time nondeterministic computation, that is, the set of polynomial-size generators equals NP/poly (see [27]). It is easy to see that for generators the equivalence problem is Π_2^{p}-complete. We will prove an analogue of Rice's Theorem for generators, even the proof is analogous.

Theorem 6.2. *Let A be a set of Boolean functions which not only depends on arity. Then* SAT *or the complement of* SAT *is p-m-reducible to the following problem: Given a generator, does the Boolean function described by it belong to A ?*

Proof. Because A does not only depend on arity there is an arity n such that one Boolean function of that arity is in A and another one is not in A. Assume as the first case that A does not contain the n-ary constant-0-function and let $f = f(y_1, \ldots, y_n)$ be a circuit which describes an n-ary Boolean function in A. We give an p-m-reduction of SAT to the set of generators which compute a Boolean function in A. Let a circuit $c = c(x_1, \ldots, x_m)$ be given. Construct the generator g with $m + n$ inputs $x_1, \ldots, x_m, y_1, \ldots, y_n$ and $n + 1$ outputs $i, r_1, \ldots, r_n$ the following way:

$$i = c(x_1, \ldots, x_m) \wedge f(y_1, \ldots, y_n) \quad \text{and} \quad r_j = y_j \text{ for } j = 1, \ldots, n.$$

If c is satisfiable, then g computes the Boolean function described by f; otherwise g computes the n-ary constant-0-function. Assume, as the second case, that A does contain the n-ary constant-0-function and let $f = f(y_1, \ldots, y_n)$ be a circuit which describes an n-ary Boolean function not in A. The same construction of a generator g from above is a p-m-reduction of the complement of SAT to the set of generators which compute a Boolean function in A. □

Remark. The restriction for A of being dependent on the arity is necessary because for example for the set of Boolean functions with odd arity the decision problem in question is polynomial-time computable.

If a problem is not decidable one still can have hope that it is recursively enumerable. The following extension of Rice's Theorem has a criterion for semantic properties of programs to be not even recursively enumerable.

Theorem 6.3 (Rice [24, Theorem 6]). *Let A be a set of partial recursive functions. If there exist two functions f, g such that f is contained in A, g is not contained in A and $f \leq g$ (that is if $f(n)$ terminates, then $f(n) = g(n)$), then the following problem is not recursively enumerable: Given a program p, does it compute a function from A ?*

We can state the analogue of the above theorem. Also the proof is analogous.

Theorem 6.4. *Let A be a set of Boolean functions with two Boolean functions $F(x_1, \ldots, x_n)$, $G(x_1, \ldots, x_n)$ such that F is contained in A, G is not contained in A and $F(a_1, \ldots, a_n) \leq G(a_1, \ldots, a_n)$ for all assignments $(a_1, \ldots, a_n)$. Then – under the assumption that* NP *is not equal* co-NP *– the following problem is not in* NP*: Given a generator, does the Boolean function computed by it belong to A ?*

Proof. Let $f(y_1, , \ldots, y_n)$ be a circuit for F and let $g(y_1, , \ldots, y_n)$ be a circuit for G. We give a p-m-reduction from the complement of SAT to the set of generators which compute a Boolean function in A. Let a circuit $c = c(x_1, \ldots, x_m)$ be given. Construct the generator h with $m+n$ inputs $x_1, \ldots, x_m, y_1, \ldots, y_n$ and $n+1$ outputs $i, r_1, \ldots, r_n$ the following way:

$$i = f(y_1, \ldots, y_n) \vee (c(x_1, \ldots, x_m) \wedge g(y_1, \ldots, y_n)), \quad \text{and} \quad r_j = y_j \text{ for } j = 1, \ldots, n.$$

If c is not satisfiable, then h computes the Boolean function F, and otherwise h computes the Boolean function G. □

References

[1] Agrawal, M., and T. Thierauf, The Boolean isomorphism problem. 37th Symposium on Foundations of Computer Science (FOCS), 1996, pp. 422 – 430.

[2] Beigel, R., A structural theorem that depends quantitatively on the complexity of SAT. Proc. 2nd Annual Conference on Structure in Complexity Theory, 1987, pp. 28 – 32.

[3] Beigel, R., NP-hard sets are P-superterse unless R = NP. Technical Report 88-04, John Hopkins University, Baltimore, 1988.

[4] Beigel, R., M. Kummer, and F. Stephan, Approximable sets. Information and Computation **120** (1995), 304 – 314.

[5] Berthiaume, A., and G. Brassard, The quantum challenge to structural complexity theory. Proc. 7th Annual Conference on Structure in Complexity Theory, 1992, pp. 132 – 137.

[6] Blass, A., and Y. Gurevich, On the unique satisfiability problem. Information and Control **55** (1982), 80 – 88.

[7] Book, R. V., J. H. Lutz, and K. Wagner, An observation on probability versus randomness with applications to complexity classes. Math. Syst. Theory **27** (1994), 201 – 209.

[8] Borchert, B., and A. Lozano, Succinct circuit representations and leaf language classes are basically the same concept. Information Processing Letters **58** (1996), 211 – 215.

[9] B. Borchert, D. Ranjan, and F. Stephan, The computational complexity of some classical equivalence relations on Boolean functions. Theory of Computing Systems **31** (1998), 679 – 693.

[10] Case, J., Effectivizing inseparability. Z. Math. Logik Grundlagen Math. **37** (1991), 97 – 111.

[11] Cook, S. A., The complexity of theorem proving procedures. Proc. 3rd Annual ACM Symposium on the Theory of Computing (STOC), 1971, pp. 151 – 158.

[12] Fellows, M. R., and N. Koblitz, Self-witnessing polynomial-time complexity and prime factorization. Proc. 7th Annual Conference on Structure in Complexity Theory, 1992, pp. 107 – 110.

[13] Fenner, S. A., L. J. Fortnow, and S. A. Kurtz, Gap-definable counting classes. J. Computer Systems Sciences **48** (1994), 116 – 148.

[14] Grollmann, J., and A. L. Selman, Complexity measures for public-key cryptosystems. SIAM J. Computing **17** (1988), 309 – 335.

[15] Gundermann, T., N. A. Nasser, and G. Wechsung, A survey on counting classes. Proc. 5th Annual Conference on Structure in Complexity Theory, 1990, pp. 140 – 153.

[16] Gupta, S., The power of witness reduction. Proc. 6th Annual Conference on Structure in Complexity Theory, 1991, pp. 43 – 59.

[17] Hartmanis, J., and L. A. Hemachandra, Complexity classes without machines: On complete languages for UP. Theoret. Computer Science **58** (1988), 129 – 142.

[18] Hemaspaandra, L. A., and J. Rothe, A second step towards circuit complexity theoretic analogs of Rice's Theorem. Proc. 23rd International Symposium on Mathematical Foundations of Computer Science, 1998, pp. 418 – 426. (A journal version will appear in Theoret. Computer Science.)

[19] Kozen, D., Indexings of subrecursive classes. Theoret. Computer Science **11** (1980), 277 – 301.

[20] Ladner, R., The circuit value problem is log-space complete for P. SIGACT News **7** (1975), 18 – 20.

[21] Ogiwara, M., and L. A. Hemachandra, A complexity theory of feasible closure properties. Proc. 6th Annual Conference on Structure in Complexity Theory, 1991, pp. 16 – 29.

[22] Papadimitriou, Ch., Computational Complexity. Addison Wesley, Reading (Mass.) 1994.

[23] Rackoff, C., Relativized questions involving probablistic algorithms. J. Ass. Computing Machinery **39** (1992), 261 – 268.

[24] Rice, H. G., Classes of recursively enumerable sets and their decision problems. Trans. Amer. Math. Soc. **74** (1953), 358 – 366.

[25] Rice, H. G., On completely recursively enumerable classes and their key arrays. J. Symbolic Logic **21** (1956), 304 – 341.

[26] Royer, J., A Connotational Theory of Program Structure. Lecture Notes in Computer Science **273**, Springer-Verlag, Berlin-Heidelberg-New York 1987.

[27] Schöning, U., On small generators. Theoret. Computer Science **34** (1984), 337 – 341.

[28] Valiant, L. G., The relative complexity of checking and evaluating. Information Processing Letters **5** (1976), 20 – 23.

[29] Valiant, L. G., and V. V. Vazirani, NP is as easy as detecting unique solutions. Theoret. Computer Science **47** (1986), 85 – 93.

[30] Veith, H., Succinct representations, leaf languages and projection reductions. Proc. 11th Annual Conference on Structure in Complexity Theory, 1996, pp. 118 – 126.

[31] Yap, C. K., Some consequences of non-uniform conditions on uniform classes. Theoret. Computer Science **27** (1983), 287 – 300.

(Received: August 7, 1999; Revised: September 15, 1999)