ELSEVIER

Theoretical Computer Science 244 (2000) 205–217

Theoretical Computer Science

www.elsevier.com/locate/tcs

# of Rice's Theorem

Lane A. Hemaspaandra [a,*,1], Jörg Rothe [b,2]

[a] *Department of Computer Science, University of Rochester, Rochester, NY 14627, USA*
[b] *Institut für Informatik, Friedrich-Schiller-Universität Jena, 07740 Jena, Germany*

Received December 1997; revised October 1998
Communicated by O.H. Ibarra

Abstract

Rice's Theorem states that every nontrivial language property of the recursively enumerable sets is undecidable. Borchert and Stephan (1997) initiated the search for complexity-theoretic analogs of Rice's Theorem. In particular, they proved that every nontrivial counting property of circuits is UP-hard, and that a number of closely related problems are SPP-hard.

The present paper studies whether their UP-hardness result itself can be improved to SPP-hardness. We show that their UP-hardness result cannot be strengthened to SPP-hardness unless unlikely complexity class containments hold. Nonetheless, we prove that every P-constructibly bi-infinite counting property of circuits is SPP-hard. We also raise their general lower bound from unambiguous nondeterminism to constant-ambiguity nondeterminism.

Keywords: Circuits; Computational complexity; Counting properties; Rice's Theorem

1. Introduction

Rice's Theorem ([28, 29]; see also [4]) states that every nontrivial language property of the recursively enumerable sets is either RE-hard or coRE-hard and thus is certainly undecidable (a corollary that itself is often referred to as Rice's Theorem).

* Corresponding author.
E-mail address: lane@cs.rochester.edu (L.A. Hemaspaandra).

[1] Supported in part by grants NSF-CCR-9322513, NSF-INT-9513368/DAAD-315-PRO-fo-ab and NSF-INT-9815095/DAAD-315-PPP-gü-ab. Work done in part while visiting Friedrich-Schiller-Universität Jena.

[2] Supported in part by grants NSF-INT-9513368/DAAD-315-PRO-fo-ab and NSF-INT-9815095/DAAD-315-PPP-gü-ab. Work done in part during a postdoctoral year at the University of Rochester, supported by a NATO Postdoctoral Science Fellowship from the Deutscher Akademischer Austauschdienst ("Gemeinsames Hochschulsonderprogramm III von Bund und Ländern").

0304-3975/00/$ - see front matter
PII: S0304-3975(98)00344-2

Theorem 1 (Rice's Theorem, Version I). *Let $\mathbb{A}$ be a nonempty proper subset of the class of recursively enumerable sets. Then either the halting problem or its complement many-one reduces to the problem*: *Given a Turing machine M, is $L(M) \in \mathbb{A}$?*

Corollary 2 (Rice's Theorem, Version II). *Let $\mathbb{A}$ be a nonempty proper subset of the class of recursively enumerable sets. Then the following problem is undecidable*: *Given a Turing machine M, is $L(M) \in \mathbb{A}$?*

Rice's Theorem conveys quite a bit of information about the nature of programs and their semantics. Programs are completely nontransparent. One can (in general) decide *nothing* – emptiness, nonemptiness, infiniteness, etc. – about the languages of given programs other than the trivial fact that each accepts some language and that language is a recursively enumerable language.[3] Recently, Kari [21] has proven, for cellular automata, an analog of Rice's Theorem: All nontrivial properties of limit sets of cellular automata are undecidable.

A bold and exciting paper of Borchert and Stephan [4] proposes and initiates the search for *complexity-theoretic* analogs of Rice's Theorem. Borchert and Stephan note that Rice's Theorem deals with properties of programs, and they suggest as a promising complexity-theoretic analog properties of Boolean circuits. In particular, they focus on counting properties of circuits. Let $\mathbb{N}$ denote $\{0,1,2,\ldots\}$. Boolean functions are functions that for some n map $\{0,1\}^n$ to $\{0,1\}$. Circuits built over Boolean gates (and encoded in some standard way – in fact, for simplicity of expression, we will often treat a circuit and its encoding as interchangeable) are ways of representing Boolean functions. As Borchert and Stephan point out, the parallel is a close one. Programs are concrete objects that correspond in a many-to-one way with the semantic objects, languages. Circuits (encoded into Σ^*) are concrete objects that correspond in a many-to-one way with the semantic objects, Boolean functions. Given an arity n circuit c, $\#(c)$ denotes under how many of the 2^n possible input patterns c evaluates to 1.

Definition 3. (1) [4] Each $A \subseteq \mathbb{N}$ is a *counting property of circuits*. If $A \neq \emptyset$, we say it is a nonempty property, and if $A \neq \mathbb{N}$, we say it is a proper property.

[3] One must stress that Rice's Theorem refers to the languages accepted by the programs (Turing machines) rather than to machine-based actions of the programs (Turing machines) – such as whether they run for at least seven steps on input 1776 (which is decidable) or whether for some input they do not halt (which is not decidable, but Rice's Theorem does not speak directly to this issue, that is, Rice's Theorem does not address the computability of the set $\{M \mid \text{there is some input } x \text{ on which } M(x) \text{ does not halt}\}$).

We mention in passing a related research line about "independence results in computer science". That line started with work of Hartmanis and Hopcroft [16] based on the nontransparency of machines, and has now reached the point where it has been shown, by Regan, that for each fixed recursively axiomatizable proof system there is a *language* with certain properties that the system cannot prove, no matter how the language is represented in the system (say, by a Turing machine accepting it). For instance, for each fixed recursively axiomatizable proof system there is a low-complexity language that is infinite, but for no Turing machine accepting the language can the proof system prove that that Turing machine accepts an infinite language. See [26, 27] and the references therein.

(2) [4] Let A be a counting property of circuits. The *counting problem for* A, $\texttt{Counting}(A)$, is the set of all circuits c such that $\#(c) \in A$.

(3) (see Garey and Johnson [10]) For each complexity class $\mathbb{C}$ and each set $B \subseteq \Sigma^*$, we say B is $\mathbb{C}$-*hard* if $(\forall L \in \mathbb{C})\,[L \leqslant_{\mathrm{T}}^{\mathrm{p}} B]$, where as is standard $\leqslant_{\mathrm{T}}^{\mathrm{p}}$ denotes polynomial-time Turing reducibility.

(4) (following usage of [4]) Let A be a counting property and let $\mathbb{C}$ be a complexity class. By convention, we say that counting property A is $\mathbb{C}$-*hard* if the counting problem for A, $\texttt{Counting}(A)$, is $\mathbb{C}$-hard. (Note in particular that by this we do not mean $\mathbb{C} \subseteq \mathrm{P}^A$ – we are speaking just of the complexity of A's counting problem.)

For succinctness and naturalness, and as it introduces no ambiguity here, throughout this paper we use "counting" to refer to what Borchert and Stephan originally referred to as "absolute counting". For completeness, we mention that their sets $\texttt{Counting}(A)$ are not entirely new: For each A, $\texttt{Counting}(A)$ is easily seen (in light of the fact that circuits can be parsimoniously simulated by Turing machines, which themselves, as per the references cited in the proof of Theorem 9, can be parsimoniously transformed into Boolean formulas) to be many-one equivalent to the set, known in the literature as SAT_A or A-SAT, $\{f \mid \text{the number of satisfying assignments to Boolean formula } f \text{ is an integer contained in the set } A\}$ [5, 15]. Thus, $\texttt{Counting}(A)$ inherits the various properties that the earlier papers on SAT_A established for SAT_A, such as completeness for certain counting classes. We will at times draw on this earlier work to gain insight into the properties of $\texttt{Counting}(A)$.

The results of Borchert and Stephan that led to the research reported on in the present paper are the following. Note that Theorem 4 is a partial analog of Theorem 1,[4] and Corollary 5 is a partial analog of Corollary 2. $\mathrm{UP} \bigoplus \mathrm{coUP}$ denotes $\{A \oplus B \mid A \in \mathrm{UP} \wedge B \in \mathrm{coUP}\}$, where $A \bigoplus B = \{0x \mid x \in A\} \cup \{1y \mid y \in B\}$.

Theorem 4 (Borchert and Stephan [4], see also the comments at the start of the proof of Theorem 9). *Let A be a nonempty proper subset of $\mathbb{N}$. Then one of the following three classes is $\leqslant_{\mathrm{m}}^{\mathrm{p}}$-reducible to $\texttt{Counting}(A)$:* NP, coNP, *or* $\mathrm{UP} \bigoplus \mathrm{coUP}$.

Corollary 5 (Borchert and Stephan [4], see also the comments at the start of the proof of Theorem 9). *Every nonempty proper counting property of circuits is* UP-*hard.*

Borchert and Stephan's paper proves a number of other results – regarding an artificial existentially quantified circuit type yielding NP-hardness, definitions and results about counting properties over rational numbers and over $\mathbb{Z}$, and so on – and we highly commend their paper to the reader. They also give a very interesting motivation. They

[4] Passing on a comment from an anonymous referee, we mention that the reader may want to also compare the $\mathrm{UP} \bigoplus \mathrm{coUP}$ occurrence in Borchert and Stephan [4] with the so-called Rice–Shapiro theorem (see, e.g., [27, 30]). We mention that in making such a comparison one should keep in mind that the Rice–Shapiro theorem deals with showing *non-membership* in RE and coRE, rather than with showing many-one *hardness* for those classes.

show that, in light of the work of Valiant and Vazirani [35], any nontrivial counting property of circuits is hard for either NP or coNP, with respect to *randomized reductions*. Their paper and this one seek to find to what extent or in what form this behavior carries over to deterministic reductions.

The present paper makes the following contributions. First, we extend the above-stated results of Borchert and Stephan, Theorem 4 and Corollary 5. Regarding the latter, from the same hypothesis as their Corollary 5 we derive a stronger lower bound – $\mathrm{UP}_{\mathrm{O}(1)}$-hardness. That is, we raise their lower bound from *unambiguous* nondeterminism to *low-ambiguity* nondeterminism. Second, we show that our improved lower bound cannot be further strengthened to SPP-hardness unless an unlikely complexity class containment – $\mathrm{SPP} \subseteq \mathrm{P}^{\mathrm{NP}}$ – occurs. Third, we nonetheless under a very natural hypothesis raise the lower bound on the hardness of counting properties to SPP-hardness. The natural hypothesis strengthens the condition on the counting property to require not merely that it is nonempty and proper, but also that it is infinite and coinfinite in a way that can be certified by polynomial-time machines.

2. The complexity of counting properties of circuits

All the notations and definitions in this paragraph are standard in the literature. Fix the alphabet $\Sigma = \{0,1\}$. FP denotes the class of polynomial-time computable functions from Σ^* to Σ^*. Given any two sets $A, B \subseteq \Sigma^*$, we say A polynomial-time many-one reduces to B $(A \leqslant_{\mathrm{m}}^{\mathrm{p}} B)$ if $(\exists f \in \mathrm{FP})(\forall x \in \Sigma^*)$ $[x \in A \Leftrightarrow f(x) \in B]$. For each set A, $\|A\|$ denotes the number of elements in A. The length of each string $x \in \Sigma^*$ is denoted by $|x|$. We use DPTM (respectively, NPTM) as a shorthand for deterministic polynomial-time Turing machine (nondeterministic polynomial-time Turing machine). Turing machines and their languages (with or without oracles) are denoted as is standard, as are complexity classes (with or without oracles), e.g., M, M^A, $L(M)$, $L(M^A)$, P, and P^A. We allow both languages and functions to be used as oracles. In the latter case, the model is the standard one, namely, when query q is asked to a function oracle f the answer is $f(q)$. For each $k \in \mathbb{N}$, the notation "$[k]$" denotes a restriction of at most k oracle questions (in a sequential – i.e., "adaptive" or "Turing" – fashion). For example, $\mathrm{P}^{\mathrm{FP}[2]}$ denotes $\{L \mid (\exists \mathrm{DPTM} M)(\exists f \in \mathrm{FP})$ $[L = L(M^f) \wedge (\forall x \in \Sigma^*)$ $[M^f(x)$ makes at most two oracle queries]]$\}$, which happens to be merely an ungainly way of describing the complexity class P. The notation "[O(1)]" denotes that, for some constant k, a "$[k]$" restriction holds.

We will define, in a uniform way via counting functions, some standard ambiguity-limited classes and counting classes. To do this, we will take the standard "#" operator ([32] for the concept and [36] for the notation, see the discussion in [19]) and will make it flexible enough to describe a variety of types of counting functions that are well-motivated by existing language classes. In particular, we will add a general restriction on the maximum value it can take on. (For the specific case of a polynomial restriction

such an operator, $\#_{\text{few}}$, was already introduced by Hemaspaandra and Vollmer [19], see below).

Definition 6. For each function $g : \mathbb{N} \to \mathbb{N}$ and each class $\mathbb{C}$, define $\#_g \cdot \mathbb{C} = \{f : \Sigma^* \to \mathbb{N} | (\exists L \in \mathbb{C})\ (\exists \text{ polynomial } s)\ (\forall x \in \Sigma^*)\ [f(x) \leqslant g(|x|) \wedge \|\{y \mid |y| = s(|x|) \wedge \langle x, y\rangle \in L\}\| = f(x)]\}$.

Note that for the very special case of $\mathbb{C} = \mathrm{P}$, which is the case of importance in the present paper, this definition simply yields classes that speak about the number of accepting paths of Turing machines that obey some constraint on their number of accepting paths. In particular, the following clearly holds for each $g : \#_g \cdot \mathrm{P} = \{f : \Sigma^* \to \mathbb{N} \mid (\exists \text{ NPTM } N)\ (\forall x \in \Sigma^*)\ [N(x) \text{ has exactly } f(x) \text{ accepting paths and } f(x) \leqslant g(|x|)]\}$.

In using Definition 6, we will allow a bit of informality regarding describing the functions g. For example, we will write $\#_1$ when formally we should write $\#_{\lambda x.1}$, and so on in similar cases. Also, we will now define some versions of the $\#_g$ operator that focus on collections of bounds of interest to us.

Definition 7. (1) For each class $\mathbb{C}$, $\#_{\text{const}} \cdot \mathbb{C} = \{f : \Sigma^* \to \mathbb{N} \mid (\exists k \in \mathbb{N})\ [f \in \#_k \cdot \mathbb{C}]\}$.

(2) (Hemaspaandra and Vollmer [19]) For each class $\mathbb{C}$, $\#_{\text{few}} \cdot \mathbb{C} = \{f : \Sigma^* \to \mathbb{N} \mid (\exists \text{ polynomial } s)\ [f \in \#_s \cdot \mathbb{C}]\}$.

Definition 8. (1) [34] $\#\mathrm{P} = \{f : \Sigma^* \to \mathbb{N} \mid (\exists \text{ NPTM } N)\ (\forall x \in \Sigma^*)\ [N(x) \text{ has exactly } f(x) \text{ accepting paths}]\}$.

(2) [33] $\mathrm{UP} = \{L \mid (\exists f \in \#_1 \cdot \mathrm{P})\ (\forall x \in \Sigma^*)\ [x \in L \Leftrightarrow f(x) > 0]\}$.

(3) ([2], see also [37]) For each $k \in \mathbb{N} - \{0\}$, $\mathrm{UP}_{\leqslant k} = \{L \mid (\exists f \in \#_k \cdot \mathrm{P})\ (\forall x \in \Sigma^*)\ [x \in L \Leftrightarrow f(x) > 0]\}$.

(4) ([20], see also [2]) $\mathrm{UP}_{\mathrm{O}(1)} = \{L \mid (\exists f \in \#_{\text{const}} \cdot \mathrm{P})\ (\forall x \in \Sigma^*)\ [x \in L \Leftrightarrow f(x) > 0]\}$. (Equivalently, $\mathrm{UP}_{\mathrm{O}(1)} = \bigcup_{k \geqslant 1} \mathrm{UP}_{\leqslant k}$.)

(5) [1] $\mathrm{FewP} = \{L \mid (\exists f \in \#_{\text{few}} \cdot \mathrm{P})\ (\forall x \in \Sigma^*)\ [x \in L \Leftrightarrow f(x) > 0]\}$.

(6) [7] $\mathrm{Few} = \mathrm{P}^{(\#_{\text{few}} \cdot \mathrm{P})\ [1]}$.

(7) $\mathrm{Const} = \mathrm{P}^{(\#_{\text{const}} \cdot \mathrm{P})\ [\mathrm{O}(1)]}$.[5]

(8) [8, 24] $\mathrm{SPP} = \{L \mid (\exists f \in \#\mathrm{P})\ (\exists g \in \mathrm{FP})\ (\forall x \in \Sigma^*)\ [(x \notin L \Leftrightarrow f(x) = 2^{|g(x)|}) \wedge (x \in L \Leftrightarrow f(x) = 2^{|g(x)|} + 1)]\}$.

It is well known that $\mathrm{UP} = \mathrm{UP}_{\leqslant 1} \subseteq \mathrm{UP}_{\leqslant 2} \subseteq \cdots \subseteq \mathrm{UP}_{\mathrm{O}(1)} \subseteq \mathrm{FewP} \subseteq \mathrm{Few} \subseteq \mathrm{SPP}$ (the final containment is due to Köbler et al. [22], see also [8] for a more general result), and clearly $\mathrm{UP}_{\mathrm{O}(1)} \subseteq \mathrm{Const} \subseteq \mathrm{Few}$. SPP plays a central role in much of complexity theory (see [9]), and in particular is closely linked to the closure properties of #P [24]. Regarding relationships with the polynomial hierarchy, $\mathrm{P} \subseteq \mathrm{UP} \subseteq \mathrm{FewP} \subseteq \mathrm{NP}$, and $\mathrm{Few} \subseteq \mathrm{P}^{\mathrm{FewP}}$ (so $\mathrm{Few} \subseteq \mathrm{P}^{\mathrm{NP}}$). It is widely suspected that $\mathrm{SPP} \not\subseteq \mathrm{PH}$ (where PH denotes

[5] As we will note in the proof of Theorem 9, $\mathrm{P}^{(\#_{\text{const}} \cdot \mathrm{P})\ [\mathrm{O}(1)]} = \mathrm{P}^{(\#_{\text{const}} \cdot \mathrm{P})\ [1]}$. Thus, the definition of Const is more analogous to the definition of Few than one might realize at first glance.

the polynomial hierarchy), though this is an open research question. UP, $\mathrm{UP}_{\mathrm{O}(1)}$, and FewP are tightly connected to the issue of whether one-way functions exist [1, 14, 20], and Watanabe [37] has shown that $\mathrm{P} = \mathrm{UP}$ if and only if $\mathrm{P} = \mathrm{UP}_{\mathrm{O}(1)}$.

Intuitively, UP captures the notion of unambiguous nondeterminism, FewP allows polynomially ambiguous nondeterminism and, most relevant for the purposes of the present paper, $\mathrm{UP}_{\mathrm{O}(1)}$ allows constant-ambiguity nondeterminism. Corollary 10 raises the UP lower bound of Borchert and Stephan (Corollary 5) to a $\mathrm{UP}_{\mathrm{O}(1)}$ lower bound. This result is obtained via the even stronger bound provided by Theorem 9, which itself extends Theorem 4.

Theorem 9. *Let A be a nonempty proper subset of $\mathbb{N}$. Then one of the following three classes is $\leqslant^{\mathrm{p}}_{\mathrm{m}}$-reducible to* $\mathtt{Counting}(A)$: NP, coNP, *or* Const.

Corollary 10. *Every nonempty proper counting property of circuits is* $\mathrm{UP}_{\mathrm{O}(1)}$*-hard* (*indeed, is even* $\mathrm{UP}_{\mathrm{O}(1)}$-$\leqslant^{\mathrm{p}}_{1-\mathrm{tt}}$*-hard*[6]).

Our proof applies a constant-setting technique that Cai and Hemaspaandra (then Hemachandra) [7] used to prove that $\mathrm{FewP} \subseteq \oplus\mathrm{P}$, and that Köbler et al. [22] extended to show that $\mathrm{Few} \subseteq \mathrm{SPP}$. Borchert, Hemaspaandra, and Rothe [3] have used the method to study the complexity of equivalence problems for OBDDs (ordered binary decision diagrams) and other structures.

Proof of Theorem 9. Let A be a nonempty proper subset of $\mathbb{N}$. The paper of Borchert and Stephan [4] (see Theorem 4 above) and – using different nomenclature – earlier papers [5, 15] have shown that (a) if A is finite and nonempty, then $\mathtt{Counting}(A)$ is $\leqslant^{\mathrm{p}}_{\mathrm{m}}$-hard for coNP, and (b) if A is cofinite and a proper subset of $\mathbb{N}$, then $\mathtt{Counting}(A)$ is $\leqslant^{\mathrm{p}}_{\mathrm{m}}$-hard for NP.

We will now show that if A is infinite and coinfinite, then $\mathtt{Counting}(A)$ is $\leqslant^{\mathrm{p}}_{\mathrm{m}}$-hard for Const. Actually, it is not hard to see that $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})\,[\mathrm{O}(1)]} = \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})\,[1]}$, and so we need deal just with $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})\,[1]}$.[7] The reason the just-mentioned equality holds is that since each of the constant number of questions (say v) has at most a constant number of possible answers (say w) one can by brute force accept each $\mathrm{P}^{(\#_w\cdot\mathrm{P})\,[v]}$ language via DPTMs that make at most $(w^{v+1}-1)/(w-1)$ queries in a truth-table fashion to a function (in fact, the same function) from $\#_w\cdot\mathrm{P}$. However, the same encoding argument ([7], see also [25]) that shows that bounded-truth-table access to a #P function can be replaced by one query to a #P function in fact also shows that $(w^{v+1-1})/(w-1)$-truth-table access to a $\#_w\cdot\mathrm{P}$ function can be replaced by one query to a $\#_{w^{\frac{w^{v+1}-1}{w-1}}-1}\cdot\mathrm{P}$ function.

[6] Where $\leqslant^{\mathrm{p}}_{1-\mathrm{tt}}$ as is standard denotes polynomial-time 1-truth-table reductions [23].

[7] This is a property that seems to be deeply dependent on the "const"-ness. For example, it is not known whether $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}[2]}$, and indeed it is known that if this seemingly unlikely equality holds then two complexity classes associated with self-specifying machines are equal [18].

Let B be an arbitrary set in $\mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$, and let $B \in \mathrm{P}^{(\#_{\mathrm{const}}\cdot\mathrm{P})[1]}$ be witnessed by some DPTM M that makes at most one query (and without loss of generality we assume that on each input x it in fact makes *exactly* one query) to some function $h \in \#_{\mathrm{const}} \cdot \mathrm{P}$. Let N' be some NPTM and let k be some constant such that for each string $z \in \Sigma^*$, $N'(z)$ has exactly $h(z)$ accepting paths and $h(z) \leqslant k$. Such a machine exists by the equality mentioned just after Definition 6. For each input x to M, let q_x be the single query to h in the run of $M(x)$.

We will call a nonnegative integer ℓ such that $\ell \in A$ and $\ell + 1 \notin A$ a *boundary event* (*of* A), and we will in such cases call $\ell + 1$ a *boundary shadow* (see, for comparison, [11–13]). Since A is infinite and coinfinite, note that it has infinitely many boundary events. We now define a function $g \in \#\mathrm{P}$ such that

$$(\forall x \in \Sigma^*)\,[M^h(x) \text{ accepts} \Leftrightarrow g(x) \in A]. \tag{1}$$

We will do so by mapping x for which $M^h(x)$ accepts to boundary events, and by mapping x for which $M^h(x)$ rejects to boundary shadows. To define g, we now describe an NPTM N that witnesses $g \in \#\mathrm{P}$.

On input x, N first computes the oracle query q_x of $M(x)$. Then $N(x)$ chooses $k+1$ constants $c_0, c_1, \ldots, c_k$ as follows.

$M^{\lambda z.j}(x) \in \{0,1\}$ denotes the result of the computation of $M(x)$ assuming the answer of the oracle was $h(q_x) = j$, where our convention is that $M^{\lambda z.j}(x) = 0$ stands for "reject" and $M^{\lambda z.j}(x) = 1$ stands for "accept". Let a_0 be the least boundary event of A (recall that boundary events are nonnegative integers, and thus it does make sense to speak of the least boundary event). Initially, choose

$$c_0 = \begin{cases} a_0 & \text{if } M^{\lambda z.0}(x) = 1, \\ a_0 + 1 & \text{if } M^{\lambda z.0}(x) = 0. \end{cases}$$

Successively, for $i = 1, \ldots, k$, do the following:

- Let $c_0, \ldots, c_{i-1}$ be the constants that have already been chosen. For each $i \in \mathbb{N}$, $\binom{i}{0} = 1$ as is standard. Let $b_i = \binom{i}{0}c_0 + \binom{i}{1}c_1 + \binom{i}{2}c_2 + \cdots + \binom{i}{i-1}c_{i-1}$.
- Let a_i be the least boundary event of A such that $b_i \leqslant a_i$.
- Set the constant

$$c_i = \begin{cases} a_i - b_i & \text{if } M^{\lambda z.i}(x) = 1, \\ a_i + 1 - b_i & \text{if } M^{\lambda z.i}(x) = 0. \end{cases}$$

After having chosen these constants,[8] $N(x)$ guesses an integer $j \in \{0, 1, \ldots, k\}$, and immediately splits into c_0 accepting paths if the guess was $j = 0$. For each $j > 0$ guessed, $N(x)$ nondeterministically guesses each j-tuple of distinct paths of $N'(q_x)$. On each such path of $N(x)$, where the j-tuple $(\alpha_1, \alpha_2, \ldots, \alpha_j)$ of paths of $N'(q_x)$ has been guessed,

[8] Note that as 2^{k+1} is also a constant we could alternatively simply build into the machine N a table that, for each of the 2^{k+1} behavior patterns M can have on an input (in terms of whether it accepts or rejects for each given possible answer from the oracle), states what constants $c_0, \ldots, c_k$ to use. The procedure just given would be used to decide the values of this table, which would then be hardwired into N.

$N(x)$ splits into exactly c_j accepting paths if each α_m, $1 \leqslant m \leqslant j$, is an accepting path of $N'(q_x)$. If, however, for some $1 \leqslant m \leqslant j$, α_m is a rejecting path of $N'(q_x)$, then $N(x)$ simply rejects (along the current path). This completes the description of N.

Recall that $h(q_x) \in \{0, 1, \ldots, k\}$ is the true answer of the oracle. Then, by the above construction, the number of accepting paths of $N(x)$ is

$$g(x) = c_0 + \binom{h(q_x)}{1} c_1 + \binom{h(q_x)}{2} c_2 + \cdots + \binom{h(q_x)}{h(q_x)-1} c_{h(q_x)-1} + \binom{h(q_x)}{h(q_x)} c_{h(q_x)}.$$

However, $c_{h(q_x)}$ has been chosen such that $g(x) = b_{h(q_x)} + c_{h(q_x)} = a_{h(q_x)} \in A$ if $M^h(x)$ accepts, and $g(x) = b_{h(q_x)} + c_{h(q_x)} = a_{h(q_x)} + 1 \notin A$ if $M^h(x)$ rejects. Since each a_i, $0 \leqslant i \leqslant k$, is a boundary event and each $a_i + 1$, $0 \leqslant i \leqslant k$, is a boundary shadow, this completes our proof of Eq. (1).

By the well-known observation (mentioned by Garey and Johnson [10, p. 169], see also the primary sources [31, 34]) that the many-one reductions of the Cook–Karp–Levin Theorem can be altered so as to be "parsimonious", there is a $\leqslant_{\mathrm{m}}^{\mathrm{p}}$-reduction that on input x (N is not an input to this $\leqslant_{\mathrm{m}}^{\mathrm{p}}$-reduction, but rather is hardwired into the reduction) outputs a Boolean formula $\phi_x(y_1, \ldots, y_n)$, where n is polynomial in $|x|$, such that the number of satisfying assignments of $\phi_x(y_1, \ldots, y_n)$ equals $g(x)$. Let $c_{\phi_x}(y_1, \ldots, y_n)$ denote (the representation of) a circuit for that formula. There is a DPTM implementing this formula-to-circuit transformation. Our reduction from B to $\mathtt{Counting}(A)$ is defined by $f(x) = c_{\phi_x}(y_1, \ldots, y_n)$. Clearly, f is polynomial-time computable, which together with Eq. (1) implies $B \leqslant_{\mathrm{m}}^{\mathrm{p}} \mathtt{Counting}(A)$ via f. □

Corollary 10 raised the lower bound of Corollary 5 from UP to $\mathrm{UP}_{\mathrm{O}(1)}$. It is natural to wonder whether the lower bound can be raised to SPP. This is especially true in light of the fact that Borchert and Stephan obtained SPP-hardness results for their notions of "counting problems over $\mathbb{Z}$" and "counting problems over the rationals"; their UP-hardness result for standard counting problems (i.e., over $\mathbb{N}$) is the short leg of their paper. However, we note that extending the hardness lower bound to SPP under the same hypothesis seems unlikely. Let BH denote the Boolean hierarchy [6]. It is well-known that $\mathrm{NP} \subseteq \mathrm{BH} \subseteq \mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{PH}$.

Proposition 11. *If $A \subseteq \mathbb{N}$ is finite or cofinite, then* $\mathtt{Counting}(A) \in \mathrm{BH}$.

This result needs no proof, as it follows easily from Lemma 3.1 and Theorem 3.1.1(a) of [5] (those results exclude the case $0 \in A$ but their proofs clearly apply also to that case) or from [15, Theorem 15], in light of the relationship between $\mathtt{Counting}(A)$ and SAT_A mentioned earlier in the present paper. Similarly, from earlier work one can conclude that, though for all finite and cofinite A it holds that $\mathtt{Counting}(A)$ is in the Boolean hierarchy, these problems are not good candidates for complete sets for that hierarchy's higher levels – or even its second level. In particular, from the

approach of the theorem and proof of [5, Theorem 3.1.2] (see also [15, Theorem 15]) it is not too hard to see that $(\exists B)[(\forall \text{ finite } A)[\texttt{Counting}(A) \text{ is not } \leqslant_{\mathrm{m}}^{\mathrm{p},B}\text{-hard for NP}^B] \wedge (\forall \text{ cofinite } A)[\texttt{Counting}(A) \text{ is not } \leqslant_{\mathrm{m}}^{\mathrm{p},B}\text{-hard for coNP}^B]]$.

In light of the fact that SPP-hardness means SPP-$\leqslant_{\mathrm{T}}^{\mathrm{p}}$-hardness, the bound of Proposition 11 yields the following result (one can equally well state the stronger claim that no finite or cofinite counting property of circuits is SPP-$\leqslant_{\mathrm{m}}^{\mathrm{p}}$-hard unless $\mathrm{SPP} \subseteq \mathrm{BH}$).

Corollary 12. *No finite or cofinite counting property of circuits is* SPP-*hard unless* $\mathrm{SPP} \subseteq \mathrm{P}^{\mathrm{NP}}$.

Though we have not in this paper discussed models of relativized circuits and relativized formulas to allow this work to relativize cleanly (and we do not view this as an important issue), we mention in passing that there is a relativization in which SPP is not contained in P^{NP} (indeed, relative to which SPP strictly contains the polynomial hierarchy) [9].

Corollary 12 makes it clear that if we seek to prove the SPP-hardness of counting properties, we must focus only on counting properties that are simultaneously infinite and coinfinite. Even this does not seem sufficient. The problem is that there are infinite, coinfinite sets having "gaps" so huge as to make the sets have seemingly no interesting usefulness at many lengths (consider, to take one example the set $\{i \mid (\exists j)[i = \texttt{AckermannFunction}(j,j)]\}$). Of course, in a recursion-theoretic context this would be no problem, as a Turing machine in the recursion-theoretic world is free from time constraints and can simply run until it finds the desired structure (which we will see is a boundary event). However, in the world of complexity theory we operate within (polynomial) time constraints. Thus, we consider it natural to add a hypothesis, in our search for an SPP-hardness result, requiring that infiniteness and coinfiniteness of a counting property be constructible in a polynomial-time manner.

Recall that a set of nonnegative integers is infinite exactly if it has no largest element. We will say that a set is P-constructibly infinite if there is a polynomial-time function that yields elements of the set at least as long as each given input.

Definition 13. (1) Let $B \subseteq \Sigma^*$. We say that B is P-*constructibly infinite* if

$$(\exists f \in \mathrm{FP})(\forall x \in \Sigma^*)[f(x) \in B \wedge |f(x)| \geqslant |x|].$$

(2) Let us adopt the standard bijection between Σ^* and $\mathbb{N}$ – the natural number i corresponds to the lexicographically $(i+1)$st string in $\Sigma^*: 0 \leftrightarrow \varepsilon$, $1 \leftrightarrow 0$, $2 \leftrightarrow 1$, $3 \leftrightarrow 00$, etc. If $A \subseteq \mathbb{N}$, we say that A is P-constructibly infinite if A, viewed as a subset of Σ^* via this bijection, is P-constructibly infinite according to Part 1 of this definition.

(3) If $A \subseteq \Sigma^*$ and $\overline{A}$ (or $A \subseteq \mathbb{N}$ and $\mathbb{N} - A$) are P-constructibly infinite, we will say that A is P-*constructibly bi-infinite*.

The above is our formal, type-correct definition, and is the definition we employ within our proof of Theorem 14. However, Part 2 of the definition is a bit long. Following a referee's suggestion, as an aside we mention a different, more intuitive definition that happens to yield the same class. Let us say that $A \subseteq \mathbb{N}$ belongs to NICE if there is a polynomial-time function g (mapping from $\mathbb{N}$ to $\mathbb{N}$) such that for all $n \in \mathbb{N}$ we have $g(n) \in A$ and $|g(n)| > |n|$, where both the explicit "length-of"s ($|g(n)|$ and $|n|$) and the one implicit in speaking of a "polynomial-time function" are with respect to the standard way of writing integers in binary without superfluous leading zeros. Though this definition differs from that of Part 2 of Definition 13 (e.g., the boundaries between lengths fall at different places), it in fact is not too hard to see that it does define exactly the same class; that is, NICE is exactly $\{A \subseteq \mathbb{N} \mid A \text{ is P-constructibly infinite}\}$.

Note that some languages that are infinite (respectively, bi-infinite) are not P-constructibly infinite (respectively, bi-infinite), e.g., languages with huge gaps between successive elements.

Borchert and Stephan [4] also study "counting problems over the rationals", and in this study they use a root-finding-search approach to establishing lower bounds. In the following proof, we apply this type of approach (by which we mean the successive interval contraction of the same flavor used when trying to capture the root of a function on $[a,b]$ when one knows initially that, say, $f(a)>0$ and $f(b)<0$) to counting problems (over $\mathbb{N}$). In particular, we use the P-constructibly bi-infinite hypothesis to "trap" a boundary event of $\overline{A}$.

Theorem 14. *Every* P-*constructibly bi-infinite counting property of circuits is* SPP-*hard.*

Proof. Let $A \subseteq \mathbb{N}$ be any P-constructibly bi-infinite counting property of circuits. Let L be any set in SPP. Since $L \in \mathrm{SPP}$, there are functions $f \in \#\mathrm{P}$ and $g \in \mathrm{FP}$ such that, for each $x \in \Sigma^*$: $(x \in L \Leftrightarrow f(x) = 2^{|g(x)|}+1) \wedge (x \notin L \Leftrightarrow f(x) = 2^{|g(x)|})$. Let h and $\overline{h}$ be FP functions certifying that A and $\overline{A}$ are P-constructibly infinite, in the exact sense of Part 2 of Definition 13. We will describe a DPTM N that $\leqslant^{\mathrm{p}}_{\mathrm{T}}$-reduces L to $\mathtt{Counting}(A)$. For clarity, let $\widehat{w}$ henceforth denote the natural number that in the above bijection between $\mathbb{N}$ and Σ^* corresponds to the string w. For convenience, we will sometimes view A as a subset of $\mathbb{N}$ and sometimes as a subset of Σ^* (and in the latter case we implicitly mean the transformation of A to strings under the above-mentioned bijection).

Since clearly $A \leqslant^{\mathrm{p}}_{\mathrm{m}} \mathtt{Counting}(A)$,[9] we for convenience will sometimes informally speak as if the set A (viewed via the bijection as a subset of Σ^*) is an oracle of the reduction. Formally, when we do so, this should be viewed as a shorthand for the

[9] Either one can encode a string n (corresponding to the number $\widehat{n}$ in binary) directly into a circuit c_n such that $\#(c_n) = \widehat{n}$ (which is easy to do), or one can note the following indirect transformation: Let N' be an NPTM that on input n produces exactly $\widehat{n}$ accepting paths. Using a parsimonious Cook–Karp–Levin reduction (as described earlier), we easily obtain a family of circuits $\{\widetilde{c_n}\}_{n \in \Sigma^*}$ such that, for each $n \in \Sigma^*$, $\#(\widetilde{c_n}) = \widehat{n}$.

complete $\leqslant^{\mathrm{p}}_{\mathrm{T}}$-reduction that consists of the $\leqslant^{\mathrm{p}}_{\mathrm{T}}$-reduction between L and A followed by the $\leqslant^{\mathrm{p}}_{\mathrm{m}}$-reduction between A and $\mathtt{Counting}(A)$.

We now describe the machine N. On input x, $|x|=n$, N proceeds in three steps. (As a shorthand, we will consider x fixed and will write N rather than $N^{\mathtt{Counting}(A)}(x)$.)

(1) N runs $\overline{h}$ and h on suitable inputs to find certain sufficiently large strings in $\overline{A}$ and A. In particular, let $\overline{h}(0^{|g(x)|+1})=y$. So we have $y\notin A$ and $|y|\geqslant|g(x)|+1$, and thus $\widehat{y}\geqslant 2^{|g(x)|+1}-1\geqslant 2^{|g(x)|}$. Recall that $|x|=n$. Since both $\overline{h}$ and g are in FP, there exists a polynomial p such that $|y|\leqslant p(n)$, and thus certainly $\widehat{y}<2^{p(n)+1}$. So let $h(0^{p(n)+2})=z$, which implies $z\in A$ and $|z|\geqslant p(n)+2$. Thus, $\widehat{z}\geqslant 2^{p(n)+2}-1>2^{p(n)+1}>\widehat{y}$. Since $h\in\mathrm{FP}$, there clearly exists a polynomial q such that $\widehat{z}<2^{q(n)}$. To summarize, N has found in time polynomial in $|x|$ two strings $y\notin A$ and $z\in A$ such that $2^{|g(x)|}\leqslant\widehat{y}<\widehat{z}<2^{q(n)}$.

(2) N performs a search on the interval $[\widehat{y},\widehat{z}]\subseteq\mathbb{N}$ to find some $\widehat{u}\in\mathbb{N}$ that is a boundary event of $\overline{A}$. That is, $\widehat{u}$ will satisfy: (a) $\widehat{y}\leqslant\widehat{u}\leqslant\widehat{z}$, (b) $\widehat{u}\notin A$, and (c) $\widehat{u}+1\in A$. Since $\widehat{z}<2^{q(n)}$, the search will terminate in time polynomial in $|x|$. For completeness we mention the very standard algorithm to search to find a boundary event of $\overline{A}$ (recall the comment above regarding access to A being in effect available to the algorithm):

Input $\widehat{y}$ and $\widehat{z}$ satisfying $\widehat{y}<\widehat{z}$, $\widehat{y}\notin A$, and $\widehat{z}\in A$.
Output $\widehat{u}$, a boundary event of $\overline{A}$ satisfying $\widehat{y}\leqslant\widehat{u}\leqslant\widehat{z}$.
  $\widehat{u}:=\widehat{y}$;
  while $\widehat{z}>\widehat{u}+1$ **do**
    $\widehat{a}:=\lfloor\frac{\widehat{u}+\widehat{z}}{2}\rfloor$; **if** $\widehat{a}\notin A$ **then** $\widehat{u}:=\widehat{a}$ **else** $\widehat{z}:=\widehat{a}$
  end while

(3) Now consider the #P function $e(\langle m,x\rangle)=m+f(x)$ and the underlying NPTM E witnessing that $e\in\#\mathrm{P}$. Let d_E be the parsimonious Cook–Karp–Levin reduction that on each input $\langle m,x\rangle$ outputs a circuit (representation) $\widetilde{c}_{\langle m,x\rangle}$ such that $\#(\widetilde{c}_{\langle m,x\rangle})=e(\langle m,x\rangle)$. Recall that N has already computed $\widehat{u}$ (which itself depends on x and the oracle). N, using d_E to build its query, now queries its oracle, $\mathtt{Counting}(A)$, as to whether $\widetilde{c}_{\langle\widehat{u}-2^{|g(x)|},x\rangle}\in\mathtt{Counting}(A)$, and N accepts its input x if and only if the answer is "yes". This completes the description of N.

As argued above, N runs in polynomial time. We have to show that it correctly $\leqslant^{\mathrm{p}}_{\mathrm{T}}$-reduces L to $\mathtt{Counting}(A)$. Assume $x\notin L$. Then $f(x)=2^{|g(x)|}$, and thus $e(\langle\widehat{u}-2^{|g(x)|},x\rangle)=\widehat{u}\notin A$. This implies that the answer to the query "$\widetilde{c}_{\langle\widehat{u}-2^{|g(x)|},x\rangle}\in\mathtt{Counting}(A)$?" is "no", and so N rejects x. Analogously, if $x\in L$, then $f(x)=2^{|g(x)|}+1$, and thus $e(\langle\widehat{u}-2^{|g(x)|},x\rangle)=\widehat{u}+1\in A$, and so N accepts x. □

Finally, though we have stressed ways in which hypotheses that we feel are natural yield hardness results, we mention that for a large variety of complexity classes (amongst them R, coR, BPP, PP, and FewP) one can state somewhat artificial hypotheses for A that ensure that $\mathtt{Counting}(A)$ is many-one hard for the given class. For example, if A is any set such that either $\{i\mid i$ is a boundary event of $A\}$ is

P-constructibly infinite or $\{i \mid i$ is a boundary event of $\overline{A}\}$ is P-constructibly infinite, then $\texttt{Counting}(A)$ is SPP-$\leqslant_{\mathrm{m}}^{\mathrm{p}}$-hard.

Acknowledgements

We are very grateful to Bernd Borchert, Edith Hemaspaandra, and Gerd Wechsung for helpful discussions and suggestions, to the anonymous referees for helpful suggestions, and to Lance Fortnow, Kenneth Regan, and Heribert Vollmer for helpful literature pointers and history. We thank Juris Hartmanis for commending to us the importance of finding complexity-theoretic analogs of index sets, and we commend to the reader, as Juris Hartmanis did to us, the open issue of finding a crisp complexity-theoretic analog of the recursion-theoretic work of Hartmanis and Lewis [17].

References

[1] E. Allender, R. Rubinstein, P-printable sets, SIAM J. Comput. 17 (6) (1988) 1193–1202.
[2] R. Beigel, On the relativized power of additional accepting paths, in: Proc. 4th Structure in Complexity Theory Conf. IEEE Computer Society Press, Silver Spring, MD, 1989, pp. 216–224.
[3] B. Borchert, L. Hemaspaandra, J. Rothe, Powers-of-two acceptance suffices for equivalence and bounded ambiguity problems, Technical Report TR-628, Department of Computer Science, University of Rochester, Rochester, NY, June 1996.
[4] B. Borchert, F. Stephan, Looking for an analogue of Rice's Theorem in circuit complexity theory, in: Proc. 1997 Kurt Gödel Colloquium, Lecture Notes in Computer Science, vol. 1289, Springer, Berlin, 1997, pp. 114–127.
[5] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, G. Wechsung, The Boolean hierarchy II: Applications, SIAM J. Comput. 18 (1) (1989) 95–111.
[6] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, G. Wechsung, The Boolean hierarchy I: Structural properties, SIAM J. Comput. 17 (6) (1988) 1232–1252.
[7] J. Cai, L. Hemachandra, On the power of parity polynomial time, Math. Systems Theory 23 (2) (1990) 95–106.
[8] S. Fenner, L. Fortnow, S. Kurtz, Gap-definable counting classes, J. Comput. System Sci. 48 (1) (1994) 116–148.
[9] L. Fortnow, Counting complexity, in: L. Hemaspaandra, A. Selman (Eds.), Complexity Theory Retrospective II, Springer, Berlin, 1997, pp. 81–107.
[10] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
[11] J. Goldsmith, Polynomial isomorphisms and near-testable sets, PhD thesis, University of Wisconsin–Madison, Madison, WI, January 1989.
[12] J. Goldsmith, L. Hemachandra, D. Joseph, P. Young, Near-testable sets, SIAM J. Comput. 20 (3) (1991) 506–523.
[13] J. Goldsmith, D. Joseph, P. Young, Self-reducible, P-selective, near-testable, and P-cheatable sets: The effect of internal structure on the complexity of a set, in: Proc. 2nd Structure in Complexity Theory Conf., 1987, pp. 50–59.
[14] J. Grollmann, A. Selman, Complexity measures for public-key cryptosystems, SIAM J. Comput. 17 (2) (1988) 309–335.
[15] T. Gundermann, G. Wechsung, Counting classes with finite acceptance types, Comput. Artif. Intell. 6 (5) (1987) 395–409.
[16] J. Hartmanis, J. Hopcroft, Independence results in computer science, SIGACT News 8 (4) (1976) 13–24.
[17] J. Hartmanis, F. Lewis, The use of lists in the study of undecidable problems in automata theory, J. Comput. System Sci. 5 (1) (1971) 54–66.

[18] L. Hemaspaandra, H. Hempel, G. Wechsung, Self-specifying machines, to appear in Intl. J. Found. Comput. Sci.
[19] L. Hemaspaandra, H. Vollmer, The Satanic notations: Counting classes beyond #P and other definitional adventures, SIGACT News 26 (1) (1995) 2–13.
[20] L. Hemaspaandra, M. Zimand, Strong forms of balanced immunity, Technical Report TR-480, Department of Computer Science, University of Rochester, Rochester, NY, December 1993. Revised, May 1994.
[21] J. Kari, Rice's Theorem for the limit sets of cellular automata, Theoret. Comput. Sci. 127 (2) (1994) 229–254.
[22] J. Köbler, U. Schöning, S. Toda, J. Torán, Turing machines with few accepting computations and low sets for PP, J. Comput. System Sci. 44 (2) (1992) 272–286.
[23] R. Ladner, N. Lynch, A. Selman, A comparison of polynomial time reducibilities, Theoret. Comput. Sci. 1 (2) (1975) 103–124.
[24] M. Ogiwara, L. Hemachandra, A complexity theory for feasible closure properties, J. Comput. System Sci. 46 (3) (1993) 295–325.
[25] C. Papadimitriou, S. Zachos, Two remarks on the power of counting, in: Proc. 6th GI Conf. on Theoretical Computer Science, Lecture Notes in Computer Science, vol. 145, Springer, Berlin, 1983, pp. 269–276.
[26] K. Regan, The topology of provability in complexity theory, J. Comput. System Sci. 36 (3) (1988) 384–432.
[27] K. Regan, Index sets and presentations of complexity classes, Theoret. Comput. Sci. 161 (1–2) (1996) 263–287.
[28] H. Rice, Classes of recursively enumerable sets and their decision problems, Trans. Amer. Math. Soc. 74 (1953) 358–366.
[29] H. Rice, On completely recursively enumerable classes and their key arrays, J. Symbol. Logic 21 (1956) 304–341.
[30] H. Rogers, Jr., The Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.
[31] J. Simon, On some central problems in computational complexity, PhD thesis, Cornell University, Ithaca, NY, January 1975. Available as Cornell Department of Computer Science Technical Report TR75-224.
[32] S. Toda, Computational complexity of counting complexity classes, PhD thesis, Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan, 1991.
[33] L. Valiant, The relative complexity of checking and evaluating, Inform. Process. Lett. 5 (1) (1976) 20–23.
[34] L. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[35] L. Valiant, V. Vazirani, NP is as easy as detecting unique solutions, Theoret. Comput. Sci. 47 (1986) 85–93.
[36] H. Vollmer, Komplexitätsklassen von Funktionen, PhD thesis, Institut für Informatik, Universität Würzburg, Würzburg, Germany, 1994.
[37] O. Watanabe, On hardness of one-way functions, Inform. Process. Lett. 27 (1988) 151–157.