# Cosmological Lower Bound on the Circuit Complexity of a Small Problem in Logic

Larry Stockmeyer
stock@acm.org

Albert R. Meyer
Massachusetts Institute of Technology
meyer@lcs.mit.edu

**Abstract.** An exponential lower bound on the circuit complexity of deciding the weak monadic second-order theory of one successor (WS1S) is proved. Circuits are built from binary operations, or 2-input gates, which compute arbitrary Boolean functions. In particular, to decide the truth of logical formulas of length at most 610 in this second-order language requires a circuit containing at least $10^{125}$ gates. So even if each gate were the size of a proton, the circuit would not fit in the known universe. This result and its proof, due to both authors, originally appeared in 1974 in the Ph.D. thesis of the first author. In this paper the proof is given, the result is put in historical perspective, and the result is extended to probabilistic circuits.

This paper was written while L. Stockmeyer was at the IBM Almaden Research Center, San Jose, CA.
A. R. Meyer's address is: Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.

This paper appears in the *Journal of the ACM*, Vol. 49, No. 6, 2002, pp. 753–784. Copyright © (2002) by Association for Computing Machinery, Inc. Permission to make digital or hard copies of part of all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.

# 1 Introduction

The goal of theoretical computer science, in a very general sense, is to understand the capabilities and limitations of computation. Not surprisingly, most attention has been directed towards demonstrating the capabilities. However, a true scientific understanding of capabilities can come only with an understanding of limitations. An early proof of a limitation of computation was the result of Abel and Galois in the early 1800's, that there is no finite algorithm to find the roots of the general quintic equation using only the rational arithmetic operations and root extraction. Demonstrating the limits of computation in a more general sense began in the 1930's with proofs of undecidability. This was followed by the development of complexity theory in the 1960's and proofs of "large" (exponential and larger) lower bounds in the 1970's. (More details of this history are given below.) These proofs of undecidability and large lower bounds are based on diagonalization. Both types of results are prone to the objection that in the real world we are interested in solving only a *finite* portion of a problem, for inputs up to a certain length. For an asymptotic lower bound $c^n$ on the time complexity of a problem where $c > 1$ is a constant, the lower bound may become "impractically large" only for very large input lengths $n$, if $c$ is only fractionally larger than 1. Indeed, in order to draw meaningful conclusions about computational complexity, it is essential to know at what finite point the asymptotic lower bound begins to take effect. This information often is implicit in the proofs of results of this type. But even though exponential lower bounds were known for several problems at the end of 1973, this information had not been carefully worked out for any specific problem. At that time we chose to study one problem in detail, with the goal of showing that solving the problem is practically infeasible even for reasonably small inputs. To do this we showed, for the logical theory WS1S, that deciding the truth of sentences of length at most 610 requires a Boolean circuit as large as the known universe. The proof of this result previously appeared only in the Ph.D. thesis of the first author [74]. The main purpose of this paper is to place the proof in an archived journal and describe the result in historical context, both before and after 1974.

We begin with some prior history, beginning in the 1930's. This period saw the introduction of computational models by Church, Turing, and others that seem to embody "computation" in a very general sense. One such model is the Turing machine. The halting problem was proved to be undecidable using the technique of diagonalization, and undecidability of other problems was shown by giving an effective (computable by a Turing machine) reduction from the halting problem to the other problem. As one example, Gödel's technique of arithmetization showed the undecidability of the first-order theory of integer arithmetic.

As real computers started to be built and used, attention shifted in the 1960's to the *computational complexity* of problems, that is, the amount of computational resources, such as time and memory, needed to solve the problem. A resource bound is expressed as a function of $n$, the length of the input to the device solving the problem, so that we may talk about polynomial bounds ($n^d$ for constant $d \geq 1$), exponential bounds ($c^n$ for constant $c > 1$), etc. In many ways the early development of complexity theory had parallels in decidability theory. Fundamental results from the 1960's include those of Rabin [58] and Hartmanis and Stearns [35], proving the existence of hierarchies of problems of strictly increasing complexity. These results were proved by diagonalization and paralleled results such as the undecidability of the halting problem (although the technical details were more complicated). In fact, Blum [15] explicitly considered a time-bounded version of the halting problem: informally, to decide if a given Turing machine halts in a certain amount of time. He proved that the "certain amount of time" is a lower bound (infinitely often) on the time used by any Turing machine that solves this problem. Ehrenfeucht [27], in a paper originally written and distributed in 1967, considered a bounded version of the first-order theory of integer arithmetic where all quantifiers are bounded by constants written in exponential notation (*e.g.*, $3^{2^5}$). He showed that the size of Boolean circuits that decide this theory must grow exponentially in the length of the input, and this was the first lower bound on the circuit complexity of a decision problem in logic. Although Ehrenfeucht's proof influenced us and our proof follows the same broad outline as his, the result itself left something to be desired as it dealt with an explicitly bounded version of an undecidable problem. For both bounded problems, the bound immediately implies decidability, and the proof of the lower bound on complexity parallels the proof of undecidability of the original problem.

During this period, Cobham [22] and Edmonds [26] proposed polynomial time complexity as a model for the tractable problems. A key to proofs of intractability in this sense was a complexity-theoretic version of effective reducibility. This was provided by *efficient reducibility*, as introduced by Cook [23] and Levin [44] and further developed by Karp [40], although its importance had been noted earlier by Meyer and McCreight [49]. While Meyer and McCreight suggested efficient reducibility as a tool to prove lower bounds on complexity, the work of Cook, Levin, and Karp focused on parallels to the complete problems of recursion theory, and this yielded the ground-breaking concept of NP-completeness. But because nontrivial lower bounds on the complexity of problems in NP are not known, it did not yield new lower bounds on complexity.

It was not long before the authors [50] put the hierarchy theorems and efficient reducibility together to prove exponential and larger lower bounds on the time and space complexity of "natural problems," meaning that the problems have some reasonable practical or mathematical motivation; they are not contrived to be complex. To apply the method to prove an exponential lower bound on the complexity of a problem $D$, for example, one shows that if $H$ is an arbitrary problem that can be solved in exponential time then $H$ is efficiently reducible to $D$. A hierarchy theorem states that there are such problems $H$ that *require* exponential time, and an exponential lower bound for $D$ follows. More details can be found, for example, in [4, 37, 77]. This method was later used to obtain lower bounds on the complexities of many problems from diverse areas. These include most of the classical decidable theories in logic, as well as many decidable problems in formal language theory, game theory, concurrency theory, and algebra; see [77] for a survey.

A lower bound obtained by this method typically has the following form, say for an exponential lower bound on the time complexity of a decision problem $D$: There is a constant $c > 1$ such that for any Turing machine $M$ that decides $D$ there are infinitely many inputs on which $M$ uses time at least $c^n$, where $n$ is the length of the input. The fact that any algorithm must use an excessively large amount of time *infinitely often* might be viewed as plausible evidence that any algorithm will also perform badly on inputs of reasonable size that actually arise in practice. We wanted, for at least one problem, to replace evidence by proof.

For the decision problem we chose the weak monadic second-order theory of one successor (WS1S). This seemed like a good choice, first because it was a natural, previously studied problem; for example, Büchi [19] and Elgot [28] had earlier proved that WS1S is decidable and had found close connections between this theory and finite state automata. More to the point, Meyer had shown in the Spring of 1972 (and published in [48]) that this problem is not elementary-recursive:

it cannot be solved in time bounded above by any constant number of compositions of exponential functions. This was an indication of the significant expressive power of WS1S, as compared to problems whose complexities had been shown to be merely single- or double-exponential. The language used to write formulas in "vanilla" WS1S includes first-order variables that range over $\mathbb{N}$ (the nonnegative integers), monadic second-order (set) variables that range over finite subsets of $\mathbb{N}$, the predicates "$y = x + 1$" and "$x \in S$" where $x$ and $y$ denote first-order variables and $S$ denotes a set variable, and the usual quantifiers and Boolean connectives. Writing formulas in this language is cumbersome as it omits several notations that are commonly used to write formulas. The language $\mathcal{L}$ used to write formulas in our result is enriched with some of these common notational abbreviations: decimal constants, writing 5 for $0+1+1+1+1+1$, $x+4$ for $x+1+1+1+1$, etc.; the binary relational symbols $\leq, <, =, \neq, >, \geq$ on integers; and set equality. These additional predicates are all expressible in WS1S, so the problem remains decidable. A precise definition of $\mathcal{L}$ is given in Section 4. Let EWS1S($n$) be the set of true sentences of length $n$ in $\mathcal{L}$. (We include a blank symbol in the alphabet, so that EWS1S($n$) essentially contains the true sentences of length at most $n$.)

Regarding our notion of "practically infeasible," it should first be noted that Turing machine time is not sufficient to measure the complexity of finite problems, because any finite problem can be decided by a finite state automaton within real time (time $n$). This is accomplished by coding a finite table of all the answers into the states of the automaton. Thus, for assessing the complexity of finite problems, account must be taken of the size or complexity of the device performing an algorithm as well as the time required by the algorithm. One quite general way to do this is to measure the number of basic operations on bits or the amount of logical circuitry required to decide the finite problem. The basic Boolean operations on bits are binary operations—*and*, *or*, *exclusive-or*, etc.—performed by "gates" with two inputs and one output. This output may be fanned out to serve as input to other gates in the circuit. This circuit model yields a basic measure of complexity for Boolean functions as well as finite decision problems (via appropriate encoding into Boolean vectors); precise definitions appear in Section 2. The circuit model was well known at the time, and the study of circuit complexity and variations of it has continued and expanded since [17, 25, 79]; this is discussed further below in this introduction and in Section 3.

The alphabet used for EWS1S($n$) has 63 symbols, each of which can therefore be coded into six binary digits. In particular, sentences of length 610 correspond to Boolean vectors of length $6 \cdot 610 = 3660$ bits, and this will be the number of inputs to a circuit that "accepts" the true sentences of length 610. The circuit is to have a single output line which gives the value one if and only if the input vector is the code of a true sentence of length 610. The main result can now be informally stated.

**Theorem 1.1** *If $C$ is a Boolean circuit that accepts* EWS1S(610)*, then $C$ contains more than* $10^{125}$ *gates.*

A quick calculation shows that the known universe could contain at most $10^{125}$ protons, even if they were packed tightly together.[1]

[1]Taking conservative current estimates of $10^{-15}$ m. for the diameter of a proton and 20 billion lightyears for the radius of the known universe.

Some words should be said about why we attempted to prove a result of this type, and why we think it was worth doing despite the apparent dearth of references to it.[2] As for the first "why," one reason was the all-purpose "because it was there." It seemed like the logical next step (and possibly the last step) in diagonalization-based proofs of intractability. The number $10^{125}$ was chosen so that the lower bound could be stated informally yet accurately and would be easily remembered, for example, "the computer must be as large as the universe." With this objective, we were curious to see how small the input length could be. Certainly the result would be less striking if 610 were replaced by, say, 610,000, and the technical challenge was to achieve an input length more like 610 than 610,000. As for importance, two arguments can be made. First, for someone with a technical interest in complexity theory, it provides an example (as far as we know, the only example) of justification for, as Allender [7] puts it, "... inside essentially every asymptotic lower bound in complexity theory, there hides a concrete statement about physical reality." Second, to the general scientifically-inclined person, it is a result about intractability that can be communicated without using technical language, *e.g.*, Turing machines and exponential asymptotic lower bounds. Theorem 1.1 has been used for this second purpose by Knuth [43], Osherson [53], and Stockmeyer and Chandra [78]. As further testimony to the usefulness of the result, it was used by Pohl [57] in the science fiction novel *Beyond the Blue Event Horizon* to explain why a supercomputer of the distant future cannot cope with every problem presented to it.

Turning to the history following 1974, the study of circuit complexity became an active area, with much of it motivated by the P =? NP question. To prove that P $\neq$ NP it would be enough to prove, for some problem in NP, that its circuit complexity is not polynomially bounded. Although such a proof is not in sight, two approaches have been explored. One is to prove "large," *e.g.*, $c^n$, lower bounds for restricted circuit models, with the hope of incrementally removing the restrictions. The other is to prove "small," *e.g.*, $cn$, lower bounds for the unrestricted model (the model used in Theorem 1.1), with the hope of incrementally improving the linear growth rate to super-polynomial.

An early result in the first category was done for the *monotone arithmetic* circuit model, where the inputs are viewed as indeterminates and the allowed operations are $+$ and $\times$. Schnorr [68] proved an exponential lower bound on the complexity of a polynomial derived from the (NP-complete) clique problem. That $+$ and $\times$ are not idempotent is crucial to the proof, as this severely limits the types of *useful* intermediate results that are computed within the circuit. An important advance was made by Razborov [60] and Andreev [9], who proved that certain problems in NP, including the clique problem, do not have polynomial complexity in the model of *monotone Boolean* circuits, where the allowed operations are the monotone Boolean operations *and* and *or*. These operations are idempotent; this allows a much wider class of useful intermediate results and increases the difficulty of proving lower bounds. See also [8, 17] for further discussion and later improvements to these results. Another restricted circuit model that has been widely studied is obtained by requiring that the depth of the circuit (the length of a longest input-to-output path) be bounded by a constant. Circuits are constructed from gates having unbounded fan-in (*i.e.*, arbitrary arity); this is needed so that the constant-depth restriction does not restrict the number of inputs to the circuit. One well-studied complexity class, $\mathrm{AC}^0$, is defined by further restricting constant-depth circuits to have polynomial size and to use the basic operations *not* and (unbounded fan-in) *and* and *or*. Study of $\mathrm{AC}^0$ was initiated by Ajtai [5] and Furst, Saxe, and Sipser [30], who

[2] A factor in the lack of references may be that it was never published in a conference or journal.

showed that the parity function is not in $AC^0$. The lower bounds on circuit size proved in these papers were improved to exponential by Yao [81] and Håstad [36].

But for unrestricted circuits, which may use all binary Boolean operations and have no restriction on their depth, all known lower bounds on the circuit complexities of problems in NP have the form $cn$ with $c \leq 3$ [67, 33, 54, 75, 16]. The proofs involve case analysis, and the constant $c$ has been increased by considering wider classes of cases. Not surprisingly, there has been little interest since 1984 in increasing $c$ above 3 by a more extensive case analysis.

All of the results mentioned in the preceding two paragraphs are proved by "combinatorial" methods that delve into the innards of a circuit $C$, assumed for contradiction to violate the lower bound being proved. In contrast, the "diagonalization-based" method[3] used to prove Theorem 1.1 makes no use of the internal structure of $C$; it views $C$ as a "black box."

Although the diagonalization-based method has been useful in proving lower bounds on the computational complexities of problems that have enough expressive power to efficiently encode arbitrary exponential-time computations, there is widespread belief, based in part on technical evidence, *e.g.*, [10], that this method will not help in proving that P $\neq$ NP. The NP-complete problems do not seem capable of efficiently encoding arbitrary exponential-time, or even barely nonpolynomial-time, computations. New ideas are needed.

We now outline the structure of the rest of the paper. Section 2 contains definitions of circuit complexity and some standard complexity classes that we will need in stating results. In Section 3 we describe Ehrenfeucht's argument and some results that were obtained later using refinements of it. In Section 4 we define EWS1S and prove a quantitative exponential lower bound on its circuit complexity; Theorem 1.1 is one consequence. In this section we also give an extension of the result to probabilistic circuits; for example, to decide EWS1S(614) with probability at least 2/3 of being correct, the circuit must contain at least $10^{125}$ gates. Section 5 is the conclusion.

## 2 Definitions

Let $\mathbb{N}$ denote the nonnegative integers and $\mathbb{N}^+$ denote the positive integers. Logarithms with no specified base are to the base 2.

### 2.1 Circuit complexity

There are several essentially equivalent definitions of Boolean circuits in the literature, for example, [25, 65, 79]. We use a definition based on straight-line algorithms.

Let $\Omega_{16} = \{\, g \mid g : \{0,1\}^2 \to \{0,1\} \,\}$ be the set of binary Boolean functions. Let $\Omega \subseteq \Omega_{16}$, $m \in \mathbb{N}^+$, and $t \in \mathbb{N}$. An $\Omega$-*circuit of size* $t$ *with* $m$ *inputs* is a sequence

$$U = \beta_m, \beta_{m+1}, \beta_{m+2}, \ldots, \beta_{m+t-1}$$

such that for $m \leq k \leq m+t-1$, the *step* $\beta_k = (i, j, g)$, where $i$ and $j$ are integers with $0 \leq i, j < k$ and $g \in \Omega$.

[3] Which might also be called the "Berry-paradox-based" method; see Remark 3.5 at the end of Section 3.

With each step $\beta_k$ we identify an *associated function* $\xi_k : \{0,1\}^m \to \{0,1\}$ by induction. First, for $0 \le k \le m-1$, define $\xi_k$ to be the $k^{\text{th}}$ projection,

$$\xi_k(b_0 b_1 b_2 \dots b_{m-1}) = b_k \quad \text{for all} \quad b_0 b_1 b_2 \dots b_{m-1} \in \{0,1\}^m.$$

If $m \le k \le m+t-1$ and $\beta_k = (i,j,g)$ then define

$$\xi_k(x) = g(\xi_i(x), \xi_j(x)) \quad \text{for} \quad x \in \{0,1\}^m.$$

If $f$ is a function, $f : \{0,1\}^m \to \{0,1\}^p$ for positive integers $m$ and $p$, then $U$ *computes* $f$ iff $U$ has $m$ inputs and there are integers $0 \le i_1, i_2, \dots, i_p \le m+t-1$ such that

$$f(x) = \xi_{i_1}(x)\, \xi_{i_2}(x) \, \dots \, \xi_{i_p}(x) \quad \text{for all} \quad x \in \{0,1\}^m.$$

The *circuit complexity* of $f$ (also called *combinational complexity* and *Boolean network complexity* in the literature), denoted $C(f)$, is the smallest $t$ such that there is an $\Omega_{16}$-circuit of size $t$ that computes $f$.[4] We use the shorthand *circuit* for an $\Omega_{16}$-circuit. Most of this paper concerns functions with range $\{0,1\}$. For a circuit $U$ as above, the *function computed by* $U$ is $\xi_{t+m-1}$. Let $U(x)$ denote $\xi_{t+m-1}(x)$.

For $n \in \mathbb{N}^+$, let $\mathcal{F}_n = \{\, f \mid f : \{0,1\}^n \to \{0,1\} \,\}$. Define the *maximum $n$-ary circuit complexity* $M(n)$ as

$$M(n) = \max\{\, C(f) \mid f \in \mathcal{F}_n \,\}.$$

A "counting" argument of Shannon [70] shows that $M(n) > (1-\varepsilon)2^n/n$ for each fixed $\varepsilon > 0$ and all sufficiently large $n$ (see, *e.g.*, [25, 65, 79]). Lupanov [47] showed that $\lim_{n\to\infty} M(n)/(2^n/n) = 1$. (In the sequel, we need only the rough bounds $a^n \le M(n) \le b^n$ for some constants $a, b > 1$ and all sufficiently large $n$.)

We now define circuit complexity for problems of deciding membership in sets of words. Let $\Gamma$ be a finite alphabet, and let $|\Gamma|$ denote the cardinality of $\Gamma$. We assume $|\Gamma| \ge 2$, and if $|\Gamma| = 2$ then $\Gamma = \{0,1\}$. For $x \in \Gamma^*$, let $|x|$ denote the length of $x$. A *language* (*over* $\Gamma$) is a set $L \subseteq \Gamma^*$. A *binary language* is a set $L \subseteq \{0,1\}^*$. If $|\Gamma| > 2$, an *encoding for* $\Gamma$ is a one-to-one function $h : \Gamma \to \{0,1\}^s$ where $s = \lceil \log|\Gamma| \rceil$.[5] If $\Gamma = \{0,1\}$, there is a unique *encoding for* $\Gamma$ defined by $h(0) = 0$ and $h(1) = 1$. Let $\hat{h} : \Gamma^* \to \{0,1\}^*$ be the extension of $h$. Let $L \subseteq \Gamma^*$ and $n \in \mathbb{N}^+$. Let $\mathcal{F}_{L,n}$ be the class of functions $f : \{0,1\}^{sn} \to \{0,1\}$ such that, for some encoding $h$ for $\Gamma$, for all $x \in \Gamma^n$, if $x \in L$ then $f(\hat{h}(x)) = 1$ and if $x \notin L$ then $f(\hat{h}(x)) = 0$. The *circuit complexity* of $L$ is the function $C_L : \mathbb{N}^+ \to \mathbb{N}$ such that, for each $n \in \mathbb{N}^+$, $C_L(n)$ is the minimum of $C(f)$ over all $f \in \mathcal{F}_{L,n}$.[6] Note that for $n$ fixed, $C_L(n)$ is the circuit complexity of deciding membership in the *finite* set $L \cap \Gamma^n$. For $\mathcal{B}$ a class of functions $B : \mathbb{N}^+ \to \mathbb{N}$, let CSIZE($\mathcal{B}$) denote the class of languages

[4] Of course there is no loss of generality in not allowing basic functions of one argument. For example, an inversion gate $\neg b$ can be computed as $g_{\text{NA}}(b,b)$ where $g_{\text{NA}}(v_1,v_2) = \neg(v_1 \wedge v_2)$. Similarly, adding the Boolean constants 0 and 1 "for free" as $\xi_{-2}$ and $\xi_{-1}$ can decrease the circuit complexity of $f$ by at most two, and not at all if none of the outputs of $f$ is a constant.

[5] By considering only block encodings, the exposition is somewhat simplified, and there is essentially no loss of generality.

[6] If $|\Gamma|$ is not a power of 2, the value of $f(y)$ (and the output of the circuit) does not matter for $y \notin \hat{h}(\Gamma^n)$. Requiring $f(y) = 0$ in these cases has no effect on our lower bound results.

$L$ such that, for some $B \in \mathcal{B}$, we have $C_L(n) \leq B(n)$ for all $n$. For example, CSIZE$(O(n^{O(1)}))$ is the class of languages having polynomial circuit complexity; in current terminology this class is called P/poly (as discussed further in Section 2.3).

A binary language $L$ has *maximum circuit complexity* if $C_L(n) = M(n)$ for all $n \geq 1$. A language $L$ has *exponential circuit complexity a.e.* if there is a rational constant $c > 1$ such that $C_L(n) > c^n$ for all sufficiently large $n$. A language $L$ has *polynomial circuit complexity* if there is a polynomial $p(n)$ such that $C_L(n) \leq p(n)$ for all $n$.

## 2.2 Time and space complexity

Other notions of the complexity of a language are its *time complexity* and *space complexity*; see, *e.g.*, [37] for definitions if needed. Let DTIME$(T(n))$ (respectively, DSPACE$(S(n))$) denote the class of languages accepted by deterministic multi-tape Turing machines within time $T(n)$ (respectively, space $S(n)$). For a class $\mathcal{B}$ of functions, DTIME$(\mathcal{B})$ and DSPACE$(\mathcal{B})$ are defined in analogue with the definition of CSIZE$(\mathcal{B})$ above. In particular, define

$$\mathrm{E} = \mathrm{DTIME}(2^{O(n)}) \quad \text{and} \quad \mathrm{ESPACE} = \mathrm{DSPACE}(2^{O(n)}).$$

A fundamental difference between time complexity and circuit complexity is that the former is measured on a *uniform* model (*e.g.*, Turing machines) where there is a single finite program that must work for all (infinitely many) inputs, and the latter is measured on a *nonuniform* model (circuits) where there can be a different finite program (a different circuit) for each input length. Indeed, one way to partition the subject of computational complexity is along the uniform/nonuniform boundary.

## 2.3 Connections between time and circuit complexity

The notion of circuit complexity is in some sense incomparable with time complexity because, as noted above, for each $L$ (even nonrecursive $L$) there is a constant $c$ such that $C_L(n) \leq c^n$. However, there is a basic relationship in one direction between these two notions of complexity: circuit complexity provides a related lower bound on time complexity. Savage [64] showed that if $L \in \mathrm{DTIME}(T(n))$ then $C_L(n) = O(T(n)^2)$. Pippenger and Fischer [56] improved this to $C_L(n) = O(T(n) \log T(n))$.

The two notions can be brought closer together if Turing machines are given a limited amount of "advice." The first result of this type was by the second author, Meyer (cited in [13]), who showed around 1973 that the class of languages having polynomial circuit complexity is exactly the class of languages that are polynomial-time Turing reducible to a sparse language $S$; the amount of advice is limited by the sparseness of the "oracle" language. A language $S$ is *sparse* if there is a polynomial $p(n)$ such that $S$ contains at most $p(n)$ words of length $n$, for all $n$. Schnorr [69] gave a more detailed relationship between circuit complexity and time complexity defined in terms of Turing machines with sparse oracles. Later Pippenger [55] defined the "advice" model of Turing machines: the advice for inputs of length $n$ is given to the machine as a string $\alpha_n$, which depends only on $n$. Using notation that came into use later, the class P/poly is defined as the class of languages that are accepted by deterministic Turing machines within polynomial time using polynomially-bounded advice ($|\alpha_n|$ is polynomial in $n$). The characterization of polynomial circuit complexity by P/poly is closely related to Meyer's characterization: it is easy to see that a sequence $\{\alpha_n\}_{n\geq 1}$ of polynomially-bounded advice strings can be encoded in a sparse language, and vice versa.

Instead of making Turing machines nonuniform by giving them advice, another way to bring time and circuit complexities closer together is to make circuits uniform by requiring that they be efficiently constructible by a Turing machine. The idea of uniform circuit complexity was introduced by Borodin [18] and further developed by Ruzzo [63]. We are concerned only with nonuniform circuit complexity as defined in Section 2.1. Of course, all of our lower bound results hold *a fortiori* for uniform circuit complexity.

### 2.4 Bounded alternation hierarchies

To state certain results we need a few classes of the bounded alternation hierarchies built on polynomial time and exponential time. These classes can be defined in terms of alternating Turing machines (ATM's) [21]. Rather than introduce this model, we give equivalent definitions in terms of bounded quantification over the arguments of a polynomial-time computable relation. Let $L \subseteq \Gamma^*$, $k \in \mathbb{N}^+$, and let $\mathcal{B}$ be a class of functions $B : \mathbb{N} \to \mathbb{N}$. The language $L$ belongs to the class $\Sigma_k(\mathcal{B})$ if there is a function $B \in \mathcal{B}$, a finite alphabet $\Delta$, and a relation $R(x, y_1, y_2, \ldots, y_k)$, computable in time polynomial in $|x| + |y_1| + \cdots + |y_k|$ for $x \in \Gamma^*$ and $y_1, \ldots, y_k \in \Delta^*$, such that for all $x \in \Gamma^*$,

$$x \in L \text{ iff } (\exists y_1)(\forall y_2)(\exists y_3)\ldots(Q_k y_k)[R(x, y_1, y_2, \ldots, y_k)] \tag{1}$$

where the quantifiers alternate (so $Q_k$ is $\exists$ if $k$ is odd or $\forall$ if $k$ is even) and where the $i^{\text{th}}$ quantification is over those $y_i \in \Delta^*$ with $|y_i| \leq B(|x|)$. The language $L$ belongs to $\Pi_k(\mathcal{B})$ if the complement of $L$ (*i.e.*, $\Gamma^* - L$) belongs to $\Sigma_k(\mathcal{B})$. (Equivalently, $\Pi_k(\mathcal{B})$ can be defined like $\Sigma_k(\mathcal{B})$ in terms of alternating quantifiers except that the leading quantifier is universal.) Also define $\Sigma_0(\mathcal{B}) = \Pi_0(\mathcal{B}) = \mathrm{DTIME}(\mathcal{B})$.

The classes of the *polynomial-time hierarchy* (first defined in [50] and further developed in [76]) are $\Sigma_k^p$ and $\Pi_k^p$ for $k \geq 0$, defined by $\Sigma_k^p = \Sigma_k(\mathcal{P})$ and $\Pi_k^p = \Pi_k(\mathcal{P})$ where $\mathcal{P}$ denotes the class of polynomial functions. In particular, $\Sigma_0^p = \mathrm{P}$ and $\Sigma_1^p = \mathrm{NP}$. The classes of the *exponential-time hierarchy*, $\Sigma_k^e$ and $\Pi_k^e$, are defined by $\Sigma_k^e = \Sigma_k(\mathcal{E})$ and $\Pi_k^e = \Pi_k(\mathcal{E})$ where $\mathcal{E}$ denotes the class of exponential functions, *i.e.*, $c^n$ for an arbitrary $c > 1$. Because the time to compute $R(x, y_1, \ldots, y_k)$ is polynomial in $|x| + |y_1| + \cdots + |y_k|$, once the exponential bound $B$ on the lengths of the $y$'s is fixed, the time to compute $R$ is bounded above by $c^{|x|}$ for some constant $c$.

In terms of ATM's, $\Sigma_k^p$ (respectively, $\Sigma_k^e$) is the class of languages accepted by ATM's that use polynomial (respectively, exponential) time, start in an existential state, and make at most $k - 1$ alternations from an existential state to a universal state or vice versa.

## 3 Ehrenfeucht's Argument and Refinements

In 1967 Ehrenfeucht proved an exponential lower bound on the circuit complexity of the first-order theory of $\mathbb{N}$ with addition, multiplication, and exponentiation, where all quantifiers are bounded by constants.[7] Constants are written in positional (*e.g.*, binary or decimal) notation and may be

[7]Although the result in [27] states only that the complexity of this problem is not polynomial, an exponential lower bound is implicit in the proof.

defined using exponential notation. A sentence is a formula containing no free variables. Writing sentences as words over some finite alphabet $\Gamma$, let BIA (Bounded Integer Arithmetic) be the language containing the true sentences in this logic. Obviously BIA is decidable, because all quantifiers are bounded by constants. In this section we outline Ehrenfeucht's [27] proof that BIA has exponential circuit complexity a.e., and give some results that were obtained later using a similar method. It is convenient to assume that the alphabet $\Gamma$ used to write formulas contains a blank symbol, so that $\mathrm{BIA} \cap \Gamma^n$ essentially contains the true sentences of length at most $n$, as opposed to exactly $n$.

For $0 \leq i < 2^n$, let $bin_n(i)$ be the length-$n$ binary representation of $i$. Recall $\mathcal{F}_n = \{ f \mid f : \{0,1\}^n \to \{0,1\} \}$.

For $f \in \mathcal{F}_n$, the *truth table of* $f$ is the binary word $\mathrm{tt}(f)$ of length $2^n$ defined by

$$\mathrm{tt}(f) = f(bin_n(0)) \cdot f(bin_n(1)) \cdot f(bin_n(2)) \cdots f(bin_n(2^n - 1)).$$

Define a linear order $<$ on $\mathcal{F}_n$, the *lexicographic order*, by $g < f$ iff $\mathrm{tt}(g)$ is lexicographically smaller than $\mathrm{tt}(f)$.

Ehrenfeucht's argument goes roughly as follows. Let $a > 1$ be a constant such that $a^n \leq M(n)$ for almost all $n$. Fix an $n \in \mathbb{N}^+$. Let $f_0$ be the lexicographically smallest function in $\mathcal{F}_n$ having maximum circuit complexity, *i.e.*, $C(f_0) = M(n)$, and $C(g) < M(n)$ for all $g < f_0$. Using Gödel's result that every r.e. set has an arithmetical representation [62, §14.4], it is easy to see (the details are not given in [27]) that there is a formula $\varphi(z)$ in the language of BIA such that, for all $x \in \{0,1\}^n$, the function $f_0(x) = 1$ iff $\varphi(1 \cdot x)$ is true, viewing $1 \cdot x$ as a binary numeral. Moreover, the length of $\varphi(1 \cdot x)$ is $dn$ where $d$ is a constant independent of $n$. Now a circuit of size $t$ that decides BIA on sentences of length $dn$ (using encoding $h : \Gamma \to \{0,1\}^*$) gives a circuit of size $t + O(n)$ that computes $f_0$, as follows. For an input $x \in \{0,1\}^n$, a circuit of size $O(n)$ computes the encoding via $h$ of the binary numeral $1 \cdot x$; this is then substituted for $z$ in $\varphi(z)$, which in turn is given as input to the circuit of size $t$ that decides $\mathrm{BIA} \cap \Gamma^{dn}$. Recalling that $C(f_0) = M(n) \geq a^n$ and choosing $1 < c < a^{1/d}$, it follows that the circuit complexity of BIA must be at least $c^n$ for almost all $n$ divisible by $d$. By padding with blanks, an exponential lower bound holds a.e.

Around 1973 the second author, Meyer, showed that $f_0$ can be computed by a Turing machine using exponential space. One of the referees pointed out that Sholomov [71] made a similar observation. This can be done because members of $\mathcal{F}_n$ and circuits of size $b^n$ can be represented by words of length exponential in $n$. Use $\mathrm{tt}(f)$ to represent $f$, and represent a circuit by its definition as a straightline algorithm (encoded as a word over some finite alphabet). Quantifications such as "for all $g < f_0$" are handled by exhaustive search, and only a constant number of exponential-length representations need be stored on the tape at any one time. In this way, Meyer showed the following. Recall $\mathrm{ESPACE} = \mathrm{DSPACE}(2^{O(n)})$.

**Theorem 3.1 (Meyer)** ESPACE *contains a binary language of maximum circuit complexity.*

This result allowed certain lower bounds on time complexity to be translated to lower bounds on circuit complexity. For example, in proving that $\mathrm{Th}\langle \mathbb{N}, + \rangle$, the first-order theory of $\mathbb{N}$ with addition (a.k.a. Presburger arithmetic) has time complexity double-exponential in $n$, Fischer and Rabin [29] show that every language $L$ in $\mathrm{DTIME}(2^{2^{O(n)}})$ is *poly-lin* reducible to $\mathrm{Th}\langle \mathbb{N}, + \rangle$, *i.e.*, it is reducible via a function $r$ computable in polynomial time and linear space; in particular,

$|r(x)| = O(|x|)$. Because ESPACE $\subseteq$ DTIME$(2^{2^{O(n)}})$, we can take $L$ to be a binary language of maximum circuit complexity, from which it follows easily that $\mathrm{Th}\langle \mathbb{N}, +\rangle$ has exponential circuit complexity a.e. (assuming again that the alphabet of $\mathrm{Th}\langle \mathbb{N}, +\rangle$ contains a blank symbol). Similarly, using the reduction of Meyer [48], any language in ESPACE is poly-lin reducible to WS1S, so WS1S has exponential circuit complexity a.e.

The exponential-space algorithm that computes $f_0$ uses double-exponential time, for example, to search over all members of $\mathcal{F}_n$. After the definition of the alternating Turing machine (ATM) model [21], it was clear that $f_0$ could be computed by an ATM using exponential time and a constant number of alternations.

**Theorem 3.2** $\Sigma_3^e \cap \Pi_3^e$ *contains a binary language of maximum circuit complexity.*

*Proof.* We define a binary language $L$ by an expression of the form (1) for $k = 3$. Choose the constant $b$ such that $M(n) \leq b^n$ for all $n$. Let $\mathcal{C}_n$ denote the set of circuits of size at most $b^n$. As above, we represent members of $\mathcal{F}_n$ and $\mathcal{C}_n$ by words of length $c^n$ for some constant $c$. For convenience we identify a function or circuit with its word representation.

Fix $n \geq 1$. For $f \in \mathcal{F}_n$ and $U \in \mathcal{C}_n$, let the predicate $comp(f, U)$ hold iff $f$ is the function computed by $U$. To decide $comp(f, U)$ it is enough to check, for all $x \in \{0, 1\}^n$, that $f(x) = 1$ iff $U(x) = 1$. This is an exponential ($2^n$) number of exponential-time computations, so $comp(f, U)$ can be computed in time $2^{O(n)}$. For $U \in \mathcal{C}_n$ and integer $t$ with $0 \leq t \leq b^n$, let $size(t, U)$ hold iff the size of $U$ is at most $t$; obviously, this predicate can also be computed in time $2^{O(n)}$. By definition,

$$C(f) \leq t \ \text{ iff } \ (\exists\, U \in \mathcal{C}_n)[\, size(t, U) \wedge comp(f, U)\, ].$$

Consider the predicate $Cmax(f, t)$ defined as follows: $Cmax(f, t)$ iff

$$(\forall g \in \mathcal{F}_n)[\, (C(g) \leq t) \ \wedge \ \neg(C(f) \leq t - 1) \ \wedge \ (g < f \Rightarrow C(g) \leq t - 1)\, ].$$

Note that if $t$ satisfies the first conjunct for all $g$, then $t \geq M(n)$. If $f, t$ satisfy the second conjunct, then $C(f) \geq t$, so $t \leq M(n)$. If $f, t$ satisfy the third conjunct for all $g$, then all $g$ with $g < f$ can be computed by a circuit of size $t - 1$; together with $t = M(n)$ and $C(f) \geq t$, this means that $f$ is the lexicographically smallest function in $\mathcal{F}_n$ having maximum circuit complexity. In summary, if $Cmax(f, t)$ then $t = M(n)$ and $f = f_0$. Define $L \cap \{0, 1\}^n$ by:

$$x \in L \ \text{ iff } \ (\exists f)(\exists t)[Cmax(f, t) \wedge f(x) = 1]. \tag{2}$$

So $L$ has maximum circuit complexity. By straightforward manipulation of quantifiers, the definition of $L$ in (2) can be written

$$(\exists f, t)(\forall g, U)(\exists\, U', U'')[R(x, f, t, g, U, U', U'')]$$

where $R$ is quantifier-free and can be decided in time $d^n$ for some $d$. Therefore, $L \in \Sigma_3^e$. Changing "$x \in L$" to $x \notin L$" in (2), it defines the complement $\overline{L}$ of $L$. Therefore, $\overline{L} \in \Sigma_3^e$, so $L \in \Pi_3^e$. ■

Because $\mathcal{F}_n$ and $\mathcal{C}_n$ can be restricted to contain words of length $2^n$ over some finite alphabet, the proof actually shows that there is a binary language of maximum circuit complexity in $\Sigma_3(2^n) \cap \Pi_3(2^n)$.

Theorem 3.2 can be used to show that $\mathrm{Th}\langle\mathbb{R},+\rangle$, the first-order theory of the reals with addition, has exponential circuit complexity a.e. This follows as above for $\mathrm{Th}\langle\mathbb{N},+\rangle$ because Berman [12], using methods of Fischer and Rabin [29], shows that if a language $L$ is accepted by an ATM simultaneously within $2^{O(n)}$ time and $O(n)$ alternations, then $L$ is poly-lin reducible to $\mathrm{Th}\langle\mathbb{R},+\rangle$.

We next mention some later results that used a similar proof method. Like many good ideas, Ehrenfeucht's argument has been discovered more than once. Kannan [39] showed the following.

**Theorem 3.3 (Kannan)** *For each $d \geq 1$ there is an $L \in \Sigma_2^p \cap \Pi_2^p$ such that $L \notin \mathrm{CSIZE}(O(n^d))$.*

(In other words, for all $c$, $C_L(n) > cn^d$ for infinitely many $n$.) Using an argument similar to the one in [27] and in the proof of Theorem 3.2, Kannan first proves Theorem 3.3 with $\Sigma_4^p \cap \Pi_4^p$ in place of $\Sigma_2^p \cap \Sigma_2^p$.[8] He then uses a result of Karp and Lipton [41], that $\mathrm{NP} \subseteq \mathrm{P/poly}$ implies $\Sigma_k^p = \Sigma_2^p$ for all $k \geq 2$, to finish the proof by considering two cases. First, if $\mathrm{NP} \subseteq \mathrm{P/poly}$ then, by [41], $\Sigma_4^p \cap \Pi_4^p = \Sigma_2^p \cap \Pi_2^p$. On the other hand, if there is a language $L \in \mathrm{NP}$ but $L \notin \mathrm{P/poly}$, then $L \in \mathrm{NP} \subseteq \Sigma_2^p \cap \Pi_2^p$ and $L \notin \mathrm{CSIZE}(O(n^d))$ for all $d$.

Scarpellini [66] later published a weaker version of Theorem 3.3 where $\Sigma_2^p \cap \Pi_2^p$ is replaced by $\Sigma_k^p$ for some (unspecified) $k$.

Kannan [39] also showed that a version of Theorem 3.3 holds at an exponentially higher level.

**Theorem 3.4 (Kannan)** *There is a constant $c > 1$ and an $L \in \Sigma_2^e \cap \Pi_2^e$ such that $L \notin \mathrm{CSIZE}(O(c^n))$.*

The constant $c$ in this result is less than 1.036. It is not known whether $\Sigma_2^e \cap \Pi_2^e$ contains a binary language of maximum circuit complexity.

It is an open question whether Theorem 3.3 (respectively, 3.4) can be improved by replacing $\Sigma_2^p \cap \Pi_2^p$ (respectively, $\Sigma_2^e \cap \Pi_2^e$) by a smaller class. Wilson [80] has considered this question in relativized worlds, cf. [10]. The circuit model is relativized to a binary "oracle" language $X$ by allowing circuits to use oracle gates of arbitrary arity, which output 1 or 0 depending on whether the input to the gate belongs to $X$ or not. An $r$-ary oracle gate contributes $r$ to the size of the circuit. Define $\Delta_2^{e,X} = \mathrm{E}^{\mathrm{NP}^X}$. (The class $\Delta_2^e = \mathrm{E}^{\mathrm{NP}}$ is the exponential-time analogue of $\Delta_2^p = \mathrm{P}^{\mathrm{NP}}$ in the polynomial-time hierarchy. Obviously, $\Sigma_1^e \subseteq \Delta_2^e \subseteq \Sigma_2^e \cap \Pi_2^e$.) Wilson constructs a recursive oracle $B$ such that if $L \in \Delta_2^{e,B}$ then the $B$-relativized circuit complexity of $L$ is linear in $n$. Thus, relative to some oracle, Theorem 3.3 (respectively, Theorem 3.4) cannot be improved to $L \in \Delta_2^p$ (respectively, $L \in \Delta_2^e$).

A striking result related to improving Theorem 3.4 in the real (unrelativized) world, by Impagliazzo and Wigderson [38], states that if E contains a language whose circuit complexity is exponential a.e., then $\mathrm{P} = \mathrm{BPP}$, where BPP is the class, defined by Gill [31], of languages accepted by polynomial-time probabilistic Turing machines with error probability bounded below 1/2. This equality would be significant because the obvious simulation of probabilistic computation by deterministic computation tries every possible outcome of the random choices made by the probabilistic algorithm, and this can cause an exponential blow-up in time complexity.

[8] The argument is not exactly the same, because representations of arbitrary $f \in \mathcal{F}_n$ have length exponential in $n$. Instead, for a suitable constant $c > d$, for each $n$ he considers $U_0$, the lexicographically smallest circuit of size $n^c$ that is equivalent to no circuit of size $n^{d+1}$. Then $L \cap \{0,1\}^n$ is defined by $U_0$.

**Remark 3.5 (The Berry Paradox)** Just as Gödel's Incompleteness Theorem can be viewed as a formalization of the Liar's Paradox, "This statement is false," Ehrenfeucht's argument can be viewed as a formalization of the Berry Paradox. The Berry Paradox, which was originally published by Bertrand Russell (who had been told of a similar paradox by Oxford University librarian G. Berry) has been stated in many forms; one is: "The least integer not nameable in fewer than nineteen syllables." The paradox is that this phrase names that integer using eighteen syllables. The formal version of this in Ehrenfeucht's proof is, for a word $x \in \{0,1\}^n$: "$f(x) = 1$ where $f$ is the least function in $\mathcal{F}_n$ not computable by a circuit of size $M(n)$." Clearly, a circuit that decides the truth of statements of this form for an arbitrary $x \in \{0,1\}^n$ must have size exponential in the length of the statement, given the fact (proved by a counting argument) that $M(n)$ is exponential. As described by Chaitin [20], the Berry Paradox also plays a role in program-size complexity (having various other names including Kolmogorov complexity and algorithmic information; see [46] for background). It is not surprising that the same technique was used for both circuit-size complexity and program-size complexity. A circuit can be viewed as a program to a "universal circuit simulator" that takes a description of a circuit $U$ and an input $x$ and determines the value $U(x)$. The simulator uses bounded time, in particular, polynomial in $|x|+|U|$. The definition of program-size complexity is similar, the main difference being that the program is given to a universal Turing machine with no bound on its running time.[9]

# 4 A Lower Bound on the Circuit Complexity of EWS1S

We give a quantitative lower bound on the circuit complexity of EWS1S; Theorem 1.1 is one corollary. Because our numerical lower bound depends on the language used to write formulas, we begin in Section 4.1 with a precise definition of the syntax of this language. Section 4.2 contains the statement of the lower bound (Theorem 4.1) and its proof. In Section 4.3 we give an extension of Theorem 1.1 to probabilistic circuits; these circuits can utilize random bits in their computations, but the error probability must be bounded below 1/2. Quantum circuits are briefly mentioned in Section 4.4. We note that the quantum circuit complexity of EWS1S is $c^n$ for some $c > 1$, but we have no numerical results.

## 4.1 Definition of EWS1S

A context-free grammar in BNF notation for $\mathcal{L}$, the language used to write formulas, is shown in Figure 1.

Let $\Gamma$ be the alphabet of $\mathcal{L}$, that is, the set of terminal symbols in Figure 1. Note that $|\Gamma| = 63$. If $\Phi \in \mathcal{L}$ then $|\Phi|$ denotes the length of $\Phi$ viewed as a word in $\mathcal{L}$.

In the absence of parentheses, the precedence order for logical connectives is $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ (decreasing). Binding of quantifiers to formulas takes precedence over all logical connectives. To improve readability, redundant parentheses are sometimes used to write formulas in the text; these are denoted as braces, $\{\ \}$, and are not counted in the length of formulas.

[9] Replacing "polynomial time" by "recursive time" gives other analogies, for example, between P and the recursive sets, NP and the r.e. sets, and the polynomial-time hierarchy and the Kleene arithmetical hierarchy.

$$
\begin{aligned}
\langle\text{member of } \mathcal{L}\rangle &::= \langle\text{formula}\rangle \mid \langle\text{member of } \mathcal{L}\rangle\,\# \\[1em]
\langle\text{formula}\rangle &::= \exists\,\langle\text{variable}\rangle\langle\text{formula}\rangle \mid \forall\langle\text{variable}\rangle\langle\text{formula}\rangle \mid \neg\langle\text{formula}\rangle \\
&\quad\ \mid \langle\text{formula}\rangle\langle\text{Boolean connective}\rangle\langle\text{formula}\rangle \mid (\langle\text{formula}\rangle) \mid \langle\text{atom}\rangle \\[1em]
\langle\text{atom}\rangle &::= \langle\text{term}\rangle\langle\text{order relation}\rangle\langle\text{term}\rangle \mid \langle\text{set variable}\rangle = \langle\text{set variable}\rangle \\
&\quad\ \mid \langle\text{term}\rangle \in \langle\text{set variable}\rangle \mid \langle\text{term}\rangle \notin \langle\text{set variable}\rangle \\[1em]
\langle\text{term}\rangle &::= \langle\text{integer variable}\rangle \mid \langle\text{constant}\rangle \mid \langle\text{integer variable}\rangle + \langle\text{constant}\rangle \\[1em]
\langle\text{Boolean connective}\rangle &::= \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow \\[1em]
\langle\text{order relation}\rangle &::= < \mid \leq \mid = \mid \neq \mid \geq \mid > \\[1em]
\langle\text{variable}\rangle &::= \langle\text{integer variable}\rangle \mid \langle\text{set variable}\rangle \\[1em]
\langle\text{integer variable}\rangle &::= \langle\text{integer variable}\rangle\langle\text{lower case}\rangle \mid \langle\text{lower case}\rangle \\[1em]
\langle\text{set variable}\rangle &::= \langle\text{set variable}\rangle\langle\text{upper case}\rangle \mid \langle\text{upper case}\rangle \\[1em]
\langle\text{lower case}\rangle &::= a \mid b \mid c \mid \cdots \mid p \mid q \\[1em]
\langle\text{upper case}\rangle &::= A \mid B \mid C \mid \cdots \mid P \mid Q \\[1em]
\langle\text{constant}\rangle &::= \langle\text{constant}\rangle\langle\text{digit}\rangle \mid \langle\text{digit}\rangle \\[1em]
\langle\text{digit}\rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid \cdots \mid 9
\end{aligned}
$$

Figure 1: The syntax of $\mathcal{L}$.

A formula $\varphi \in \mathcal{L}$ is a *sentence* iff it contains no free variables. Let EWS1S be the set of sentences in $\mathcal{L}$ that are true under the standard interpretation of $\mathbb{N}$, with set variables ranging over finite subsets of $\mathbb{N}$. (Leading zeroes are ignored in interpreting constants.) The symbol # denotes a blank "padding" character which is ignored in determining the truth of a sentence. Because sentences can be padded with blanks, $C_{\mathrm{EWS1S}}(n)$ measures the circuit complexity of deciding sentences of length $\leq n$.

### 4.2 The lower bound and its proof

**Theorem 4.1** *Let $k, m, n$ be positive integers such that*

1. $m > k + 1 + \log\log(2^k + m)$,
2. $k - 24 \geq 2\log m$, *and*
3. $n \geq 459 + \lfloor(\log_{10}2)m\rfloor + 11\lfloor\log_{10}m\rfloor$.

*Then* $C_{\mathrm{EWS1S}}(n) > 2^{k-3}$.

Theorem 4.1 is proved below. For a fixed numerical value of $n$, a lower bound on $C_{\mathrm{EWS1S}}(n)$ is obtained by choosing $k$ and $m$ to satisfy the above constraints. For example, we can now obtain the precise formulation of Theorem 1.1.

**Theorem 4.2** $C_{\mathrm{EWS1S}}(610) > 10^{125}$.

*Proof.* Choose $k = 420$, $m = 430$, $n = 610$, and note that $2^{416} > 10^{125}$. ■

The proof of Theorem 4.1 is along the same lines as Ehrenfeucht's proof and the proof of Theorem 3.2. The key step is Lemma 4.6, which states that if $k$, $m$, and $n$ satisfy certain constraints, then there is a function $f_0 : \{0,1\}^m \to \{0,1\}$ of "large" ( $> 2^{k-3}$) circuit complexity such that questions about the value of $f_0$ on words of length $m$ can be transformed to questions about membership of sentences of length $n$ in EWS1S; moreover, the circuit complexity of the transformation $\tau$ is relatively "small." It then follows easily that the circuit complexity of EWS1S must be almost as large as that of $f_0$. For assume that the circuit complexity of EWS1S is small. Then by placing a circuit that computes $\tau$ in series with a small circuit that accepts $\mathrm{EWS1S} \cap \Gamma^n$, we obtain a small circuit that computes $f_0$, which is a contradiction. One way to proceed with the proof would be to construct a specific exponential-space Turing machine $M$ such that $M$ accepts a language $L$ of maximum circuit complexity (Theorem 3.1), and then use the efficient reduction of Meyer [48] to reduce $L$ to EWS1S. After estimating the length of the EWS1S sentence that would result, we decided that it would be better to carry out a direct arithmetization of circuits by EWS1S formulas, following the outline of the proof of Theorem 3.1.

One preliminary result, an "abbreviation trick," is required before proving Theorem 4.1. If $\Phi$ is a logical formula involving several occurrences of a subformula $\Theta$, the trick allows $\Phi$ to be written equivalently as a formula involving only one occurrence of $\Theta$. Special cases of the trick were discovered independently by several people in the early 1970's. Here we give a fairly general version, due to M. Fischer and the second author around 1973.

In describing the trick, $b, c, p, r \in \mathbb{N}^+$, and variables may be either first-order variables or second-order (set) variables. We always apply the trick to formulas $\Phi$ of the form

$$\Phi(u_1, \ldots, u_b) = Q_1 z_1 Q_2 z_2 \ldots Q_c z_c \, \Psi(u_1, \ldots, u_b, z_1, \ldots, z_c)$$

where $Q_1, \ldots, Q_c$ are quantifiers, $u_1, \ldots, u_b$ denote variables that occur free in $\Phi$, and $z_1, \ldots, z_c$ denote variables. Here $\Psi$ denotes a formula (with free variables $u_1, \ldots, u_b, z_1, \ldots, z_c$) of the form

$$\Psi = ( \ \ldots \ \Theta(v_{1,1}, \ldots, v_{1,p}) \ \ldots \ \Theta(v_{2,1}, \ldots, v_{2,p}) \ \ldots \ \Theta(v_{r,1}, \ldots, v_{r,p}) \ \ldots \ )$$

where $\Theta(v_1, \ldots, v_p)$ denotes a formula of $p$ free variables $v_1, \ldots, v_p$, and for $1 \le i \le r$ the $i^{\text{th}}$ occurrence $\Theta(v_{i,1}, \ldots, v_{i,p})$ of $\Theta$ in $\Psi$ denotes a substitution instance of $\Theta(v_1, \ldots, v_p)$ with $v_1$ replaced by $v_{i,1}$, $v_2$ replaced by $v_{i,2}$, and so on. Each $v_{i,j}$ (for $1 \le i \le r$ and $1 \le j \le p$) denotes either a variable or a constant. In the cases we consider: each $v_{i,j}$ that is a variable is either free in $\Phi$ (it is one of $u_1, \ldots, u_b$) or is bound by one of the quantifiers $Q_1, \ldots, Q_c$ (it is one of $z_1, \ldots, z_c$); and $\Theta$ has the form $(\ldots)$ preceded by zero or more quantifiers.

Under these circumstances, $\Phi$ can be written equivalently as a formula $\Phi'$ involving one occurrence of $\Theta$ as follows. First let $\Psi'$ be the formula obtained from $\Psi$ by replacing the $i^{\text{th}}$ occurrence, $\Theta(v_{i,1}, \ldots, v_{i,p})$, of $\Theta$ by the atomic formula "$y_i = 1$" for $1 \le i \le r$, where $y_1, \ldots, y_r$ denote new variables. Now we use "dummy variables" $y, d_1, \ldots, d_p$, and write a separate formula to ensure that if $y = y_i$ and $d_j = v_{i,j}$ for some $i$ and all $j = 1, 2, \ldots, p$, then $y = 1$ iff $\Theta(d_1, \ldots, d_p)$ is true. That is:

$$\begin{aligned} \Phi'(u_1, \ldots, u_b) = Q_1 z_1 \ldots Q_c z_c \exists y_1 \ldots \exists y_r ( \, & \Psi' \\ & \wedge \forall d_1 \ldots \forall d_p \forall y ( \{ \bigvee_{i=1}^{r} \{ \, d_1 = v_{i,1} \ \wedge \ \ldots \ \wedge \ d_p = v_{i,p} \ \wedge \ y = y_i \, \} \} \\ & \qquad \Rightarrow (y = 1 \Leftrightarrow \Theta(d_1, \ldots, d_p)) \ )). \end{aligned}$$

In the cases we consider, $\Phi$ uses sufficiently few variables that the additional variables $y_1, \ldots, y_r$, $y$, $d_1, \ldots, d_p$ can each be written as a single letter. Also, each of the $v_{i,j}$ is either a single letter or a single digit.

Under these conditions, the length of $\Phi'$ is related to the lengths of $\Phi$ and $\Theta$ as follows.

**Length relation for the abbreviation trick**:

$$|\Phi'| = |\Phi| + (1 - r)|\Theta| + (4pr + 9r + 2p + 13).$$

In particular, the symbols $Q_1 z_1 \ldots Q_c z_c$ plus those symbols in $\Psi'$ contribute $(|\Phi| + 3r - r|\Theta|)$ to $|\Phi'|$.

Let NAND $= \{g_{\text{NA}}\}$ where the binary Boolean function $g_{\text{NA}}$ is defined by $g_{\text{NA}}(v_1, v_2) = \neg(v_1 \wedge v_2)$.

If $x \in \{0,1\}^m$, then $int(x)$ is the nonnegative integer $z$ such that $x$ is a reverse binary representation of $z$ (possibly with following zeroes). For example, $int(111000) = 7$ and $int(101100) = 13$. Define the *encoding* $enc(x)$ *of* $x$ by $enc(x) = m(int(x) + 1)$. Note that $enc$ is an injection from $\{0,1\}^m$ into $\mathbb{N}$.

Let $F \subset \mathbb{N}$. Then $fct(F)$ is the function mapping $\{0,1\}^m$ to $\{0,1\}$ defined by

$$fct(F)(x) = 1 \ \text{ iff } \ enc(x) \in F.$$

This is the means by which functions from $\{0,1\}^m$ to $\{0,1\}$ are encoded as finite sets of integers in our arithmetization. Note that for each $f \in \mathcal{F}_m$ there is a finite set $F$ such that $f = fct(F)$.

**Lemma 4.3** *Let $k$ and $m$ satisfy* (1) *of Theorem 4.1. There is a formula* $\textsc{Easy}(F)$ *in* $\mathcal{L}$ *(depending on $k$ and $m$) such that:*

(a) *For all finite $F \subset \mathbb{N}$, the formula* $\textsc{Easy}(F)$ *is true iff there is a* NAND*-circuit of size* $2^k$ *with $m$ inputs that computes $fct(F)$; and*

(b) $|\textsc{Easy}(F)| \leq 377 + 10\lfloor \log_{10} m \rfloor$.

*Proof.* We first write a formula $\textsc{Easy}'(F)$ involving several occurrences of a subformula, and then obtain $\textsc{Easy}(F)$ from $\textsc{Easy}'(F)$ using the abbreviation trick described above.

Some notation is helpful. If $S \subseteq \mathbb{N}$, let $seq(S)$ denote the (infinite) binary sequence $b_0 b_1 b_2 \cdots$, where $b_i = 1$ if $i \in S$ and $b_i = 0$ if $i \notin S$. Let $m\text{-}word(S,j)$ denote the finite binary subword $b_j b_{j+1} b_{j+2} \cdots b_{j+m-1}$ of $seq(S)$ (this word has length $m$).

Let $dec(m)$ denote the decimal representation of $m$. Let $dec(k)$ be a decimal representation of $k$ with leading zeroes if necessary to make $|dec(k)| = |dec(m)|$. (Constraint (1) implies $k < m$.)

The formula $\textsc{Easy}'(F)$ is a conjunction of five subformulas. The first four subformulas, $\psi_1$, $\psi_2$, $\psi_3$, and $\psi_4$, place constraints on the variables $B$, $P$, $d$, and $q$ (which are free variables in these subformulas). The last subformula $\psi_5$ expresses that $fct(F)$ is the function computed by some NAND-circuit of size at most $2^k$.

### 4.2.1 Construction of $\psi_1$

First, $\psi_1(B,d,a)$ is written so that $\forall a\,(\psi_1(B,d,a))$ is true iff $d \in B$ and $B = B_0$ where

$$B_0 = \{\, z \mid m \leq z \leq d \text{ and } z \equiv 0 \bmod m \,\}.$$

$\psi_1(B,d,a)$ is

$$\begin{aligned} & d \in B \ \wedge \ dec(m) \in B \\ & \quad \wedge \ (\{a < dec(m) \ \vee \ a > d\} \ \Rightarrow \ a \notin B) \\ & \quad \wedge \ (\{a < d \ \wedge \ a \neq 0\} \ \Rightarrow \ (a \in B \ \Leftrightarrow \ a + dec(m) \in B)). \end{aligned} \tag{$\psi_1$}$$

### 4.2.2 Construction of $\psi_2$

Assuming $B = B_0$ and $d \in B$, then $\forall a\,(\psi_2(B,P,d,a))$ is true iff $m\text{-}word(P,0) = 1^m$ and $m\text{-}word(P,mi)$ is a reverse binary representation of $(i-1) \bmod 2^m$, for all integers $i$ such that $m \leq mi \leq d$. That is, $seq(P) =$

$$\overbrace{1111\cdots11}^{m}\overbrace{0000\cdots00}^{m}\overbrace{1000\cdots00}^{m}\overbrace{0100\cdots00}^{m}\overbrace{1100\cdots00}^{m}\overbrace{0010\cdots00}^{m}\ \cdots \tag{3}$$

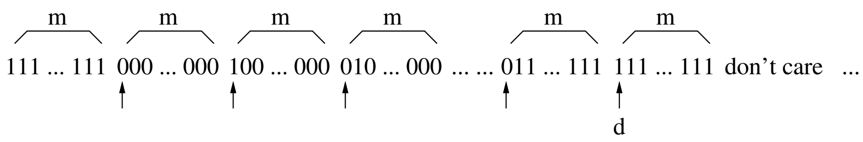



Figure 2: $P$, $B$, and $d$.

and, if $seq(P) = p_0 p_1 p_2 \cdots$ where $p_0, p_1, p_2, \ldots$ are bits, then this pattern continues at least to bit $p_{d+m-1}$ of $seq(P)$. The bits of $seq(P)$ beyond the $(d+m-1)^{\text{th}}$ are not constrained by $\psi_2$. The subformula $\psi_2$ is similar to one used by Robertson [61].

$\psi_2(B, P, d, a)$ is

$$(a < dec(m) \;\Rightarrow\; a \in P) \;\wedge\; (a < d \;\Rightarrow\; \textsc{Flip}(B, P, a)) \qquad (\psi_2)$$

where $\textsc{Flip}(B, P, a)$ iff bits $p_a$ and $p_{a+m}$ have the correct relationship, either equal or not equal, in $seq(P)$. Note that $p_a \neq p_{a+m}$ iff there is a $b \in B \cup \{0\}$ with $b \leq a$ such that, for all $i$ with $b \leq i < a$, bit $p_i = 1$. To see this, say first that $a \notin B \cup \{0\}$. If there is such a $b$, then $b = b_0$ where $b_0$ is the largest member of $B \cup \{0\}$ satisfying $b_0 < a$. So adding one to the reverse binary representation $m\text{-}word(P, b)$ will propagate a carry to bit $p_a$, thus flipping this bit. On the other hand, if such a $b$ does not exist then the carry will not propagate as far as bit $p_a$. In the case $a \in B \cup \{0\}$, there is such a $b$, namely $b = a$ (in this case there is no $i$ with $b \leq i < a$, so "for all $i$" is vacuously true); this is correct because the lowest order bit always flips. Thus $\textsc{Flip}(B, P, a)$ is

$$\begin{aligned}&((a \in P \;\Leftrightarrow\; a + dec(m) \notin P)\\ &\quad \Leftrightarrow\; \exists b\,((b \in B \;\vee\; b = 0) \;\wedge\; b \leq a \;\wedge\; \forall i(\{b \leq i \;\wedge\; i < a\} \;\Rightarrow\; i \in P))).\end{aligned}$$

#### 4.2.3 Construction of $\psi_3$

Assuming that $B = B_0$, $d \in B$, and $seq(P)$ is as in (3) where this pattern continues at least to bit $p_{d+m-1}$ of $seq(P)$, then $\forall a\,(\psi_3(P, d, a))$ is true iff $d \equiv 0 \bmod m2^m$. The subformula $\psi_3$ states simply that $m\text{-}word(P, d) = 1^m$.

$\psi_3(P, d, a)$ is

$$(\{d \leq a \;\wedge\; a < d + dec(m)\} \;\Rightarrow\; a \in P). \qquad (\psi_3)$$

Recall that $d \in B$ and $0 \notin B$ by $(\psi_1)$, and thus $d > 0$. Now writing $seq(P) = 1^m\sigma$, the formula $\forall a\,(\psi_1 \wedge \psi_2 \wedge \psi_3)$ implies that $\sigma$ cycles at least once through the $2^m$ binary words of length $m$. See Figure 2, where $seq(P)$ has been broken into blocks of length $m$ and arrows point to those positions of $seq(P)$ that belong to $B$.

#### 4.2.4 Construction of $\psi_4$

If $B$ and $P$ are as in Figure 2, then $\forall a\,(\psi_4(B, P, q, a))$ is true iff $q \in B$ and $q \leq m2^k$. We use that if $a$ is the smallest number in $B$ such that $a + k \in P$, then $a = m(2^k + 1)$.

$\psi_4(B, P, q, a)$ is

$$q \in B \;\wedge\; (\{a \in B \;\wedge\; a \leq q\} \;\Rightarrow\; a + dec(k) \notin P). \qquad (\psi_4)$$

To summarize $\psi_1$ through $\psi_4$, if $\forall a\,(\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4)$ is true then:

$$\begin{aligned}
& d \equiv 0 \bmod m2^m \quad \text{and} \quad d > 0, \\
& B = \{\, z \mid m \le z \le d \quad \text{and} \quad z \equiv 0 \bmod m \,\}, \\
& \mathit{seq}(P) \text{ is as in Figure 2,} \\
& q \le m2^k \quad \text{and} \quad q \in B.
\end{aligned} \tag{4}$$

#### 4.2.5 Construction of $\psi_5$

We first describe a formula, MATCH, that is used as a subformula in $\psi_5$. We then state the relevant properties of MATCH in Claim 4.4.

MATCH$(X_1, w_1, X_2, w_2)$ is

$$\begin{aligned}
\exists K\, \forall b\, (& w_1 < w_2 \ \wedge\ (w_1 \in B \ \vee\ w_1 < \mathit{dec}(m)) \\
& \wedge\ (b < w_1 + \mathit{dec}(m) \ \Rightarrow\ (b \in K \ \Leftrightarrow\ b \in X_1)) && (5)\\
& \wedge\ (\{w_1 \le b \ \wedge\ b < w_2\} \ \Rightarrow\ (b \in K \ \Leftrightarrow\ b + \mathit{dec}(m) \in K)) && (6)\\
& \wedge\ (w_2 \le b \ \Rightarrow\ (b \in K \ \Leftrightarrow\ b \in X_2))). && (7)
\end{aligned}$$

**Claim 4.4** *Assume that $B$, $P$, $d$, and $q$ are as in* (4)*. Let $S, S_1, S_2$ be finite subsets of $\mathbb{N}$.*

(a) *Let $z_1, z_2 \in B \cup \{0\}$.* MATCH$(S_1, z_1, S_2, z_2)$ *is true iff $z_1 < z_2$ and $m$-$\mathit{word}(S_1, z_1) = m$-$\mathit{word}(S_2, z_2)$.*

(b) *Let $i \in \mathbb{N}$ and $a \in B$.* MATCH$(P, i, S, a)$ *is true iff $i < a$ and either*

  (i) *$i \in B$ and $m$-$\mathit{word}(S, a) = m$-$\mathit{word}(P, i)$, or*

  (ii) *$0 \le i < m$ and $m$-$\mathit{word}(S, a) = 0^i 1^{m-i}$.*

(c) *Let $a \in B$ with $a \le q$. There is at most one $i \in \mathbb{N}$ such that* MATCH$(P, i, S, a)$ *is true.*

*Proof.* (a) The last three conjuncts of MATCH (holding $\forall b$) constrain $\mathit{seq}(K)$ in terms of $\mathit{seq}(S_1)$ and $\mathit{seq}(S_2)$. Conjunct (5) says that $\mathit{seq}(K)$ must match $\mathit{seq}(S_1)$ in bit positions 0 through $z_1+m-1$. Conjunct (6) says that $\mathit{seq}(K)$ must match itself, $m$ bits to the left, in bit positions $z_1+m$ through $z_2+m-1$. Conjunct (7) says that $\mathit{seq}(K)$ must match $\mathit{seq}(S_2)$ in all bit positions $\ge z_2$. Conjuncts (5), (6), and (7) are true $\forall b$ iff $m$-$\mathit{word}(S_1, z_1) = m$-$\mathit{word}(S_2, z_2)$, because it is necessary and sufficient to take, for $b \in B \cup \{0\}$, $m$-$\mathit{word}(K, b) = m$-$\mathit{word}(S_1, b)$ for $0 \le b \le z_1$, $m$-$\mathit{word}(K, b) = m$-$\mathit{word}(S_1, z_1)$ for $z_1 < b \le z_2$, and $m$-$\mathit{word}(K, b) = m$-$\mathit{word}(S_2, b)$ for $z_2 \le b$. This is illustrated in Figure 3(a) for $m = 6$; binary words are broken into blocks of length 6 for readability.

(b) By the second conjunct of MATCH, there are two cases: $i \in B$ or $i < m$. In the first case, we have $m$-$\mathit{word}(P, i) = m$-$\mathit{word}(S, a)$ by part (a). So assume that $i < m$. The (unique) choice for $\mathit{seq}(K)$ is illustrated in Figure 3(b). Formally, recall that $1^m 0^m$ is a prefix of $\mathit{seq}(P)$. Therefore, (5) is true $\forall b$ iff $1^m 0^i$ is a prefix of $\mathit{seq}(K)$. Now conjuncts (5) and (6) are both true $\forall b$ iff $m$-$\mathit{word}(K, 0) = 1^m$ and $m$-$\mathit{word}(K, b) = 0^i 1^{m-i}$ for all $b \in B$ with $b \le a$ (recall $a \in B$). Finally, conjunct (7) is true $\forall b$ iff $m$-$\mathit{word}(K, b) = m$-$\mathit{word}(S, b)$ for all $b \in B$ with $a \le b$. All of these constraints on $K$ can be met iff $m$-$\mathit{word}(S, a) = 0^i 1^{m-i}$.

$z_1 = 6$

$$
\begin{array}{rl}
\text{seq}(S_1) = & 010010\ \ 101100\ \ 000000\ \ 101111\ \ 000000\ \ 000000\ \ \ldots \\
\text{seq}(K) = & 010010\ \ 101100\ \ 101100\ \ 101100\ \ 010011\ \ 000000\ \ \ldots \\
\text{seq}(S_2) = & 100100\ \ 000100\ \ 111111\ \ 101100\ \ 010011\ \ 000000\ \ \ldots
\end{array}
$$

$z_2 = 18$

(a) How $K$ is chosen in part (a).

$i = 4$

$$
\begin{array}{rl}
\text{seq}(P) = & 111111\ \ 000000\ \ 100000\ \ 010000\ \ 110000\ \ 001000\ \ \ldots \\
\text{seq}(K) = & 111111\ \ 000011\ \ 000011\ \ 000011\ \ 010011\ \ 000000\ \ \ldots \\
\text{seq}(S) = & 100100\ \ 000100\ \ 010101\ \ 000011\ \ 010011\ \ 000000\ \ \ldots
\end{array}
$$

$a = 18$

(b) How $K$ is chosen in part (b).

Figure 3: Illustrating the proof of Claim 4.4.

(c) Fix $a \in B$ with $a \leq q$. Constraint (1) of Theorem 4.1 implies $k \leq m-1$. Now $a \leq q \leq m2^k \leq m2^{m-1}$ implies that for all $i_1, i_2 \in B$ with $i_1, i_2 < a$:

$$m\text{-}word(P, i_1) = m\text{-}word(P, i_2) \text{ iff } i_1 = i_2 \tag{8}$$

$$m\text{-}word(P, i_1) \in \{0,1\}^{m-1} \cdot 0. \tag{9}$$

Now suppose that $\textsc{Match}(P, i_1, S, a)$ and $\textsc{Match}(P, i_2, S, a)$ are both true. Part (b) of the claim implies $i_1, i_2 < a$ and one of four cases.

First, if $i_1, i_2 \in B$, then part (a) implies that $m\text{-}word(P, i_1) = m\text{-}word(S, a) = m\text{-}word(P, i_2)$. So $i_1 = i_2$ by (8).

Second, if $i_1, i_2 < m$, then part (b) implies that $0^{i_1}1^{m-i_1} = m\text{-}word(S, a) = 0^{i_2}1^{m-i_2}$, so $i_1 = i_2$.

We show that the remaining two cases, where one of $i_1, i_2$ belongs to $B$ and the other is less than $m$, cannot occur. Say that $i_1 \in B$ and $i_2 < m$, the other case being symmetric. Then $m\text{-}word(P, i_1) = m\text{-}word(S, a)$ and $i_1 < a$ because $\textsc{Match}(P, i_1, S, a)$ is true, and $m\text{-}word(S, a) = 0^{i_2}1^{m-i_2}$ because $\textsc{Match}(P, i_2, S, a)$ is true. So $m\text{-}word(P, i_1) = 0^{i_2}1^{m-i_2}$ and $m - i_2 \geq 1$. This contradicts (9), which states that $m\text{-}word(P, i_1)$ must end with 0 when $i_1 \in B$ and $i_1 < a \leq q$.

This completes the proof of Claim 4.4. ■

The next step is to describe how subsets of $\mathbb{N}$ are viewed as representing circuits and computations of circuits. It is natural to encode functions $f \in \mathcal{F}_m$ as sets and encode inputs $x \in \{0,1\}^m$ as integers. Then "$f(x) = 1$" can be expressed by a single set membership. However, encoding inputs $x$ as integers creates a problem in expressing "$U(x) = 1$" where $U$ denotes a circuit, because the computation of $U$ on $x$ requires the bits of $x$. This is handled by defining the computation of an encoded circuit on an encoded $x$ to be a finite set $D$ such that (among other properties) $x$ is a prefix of $seq(D)$. We can then express that the computation $D$ "starts correctly" by $\textsc{Match}(D, 0, P, e)$ where $e = enc(x) = m(int(x) + 1)$ is the encoding of $x$.

Let $S$ (for "small") denote $\{\, z \mid z \in \mathbb{N} \text{ and } 0 \le z < m \,\}$. Define the function $\alpha : S \cup B \to \mathbb{N}$ by

$$\alpha(a) = \begin{cases} a & \text{if } a \in S \\ a/m + m - 1 & \text{if } a \in B. \end{cases}$$

Note that $\alpha$ is strictly increasing, and $\alpha$ maps $S \cup B$ one-to-one onto the set of integers $z$ with $0 \le z \le d/m + m - 1$.

Let $B, P, d, q$ be as in (4), and let $I, J \subseteq \mathbb{N}$. The pair $(I, J)$ is *legal* iff $I$ and $J$ are finite and

$$(\forall a \in B)(\exists\, i, j \in \mathbb{N})[\,(\textsc{Match}(P, i, I, a) \;\wedge\; \textsc{Match}(P, j, J, a))\,]. \tag{10}$$

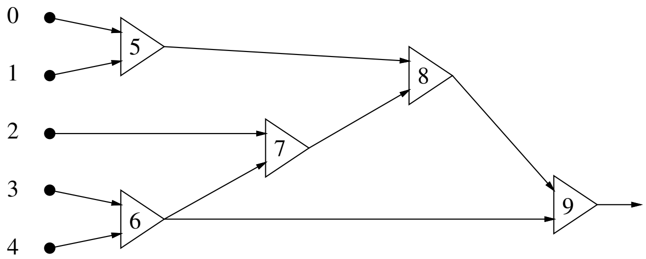
It is important to note by Claim 4.4(c) that, if $(I, J)$ is legal, then for each $a \in B$ with $a \le q$ there is a unique $i$ such that $\textsc{Match}(P, i, I, a)$ and a unique $j$ such that $\textsc{Match}(P, j, J, a)$; call these $i_a$ and $j_a$, respectively. In particular, $i_a, j_a < a$. If $(I, J)$ is legal, then $q\text{-}circuit(I, J)$ is the (unique) NAND-circuit $U$ of size $t \overset{\text{def}}{=} q/m$ with $m$ inputs, $U = \beta_m, \beta_{m+1}, \ldots, \beta_{m+t-1}$, where $\beta_{\alpha(a)} = (\alpha(i_a), \alpha(j_a), g_{\mathrm{NA}})$ for all $a \in B$ with $a \le q$. This is a legal definition of a circuit because $i_a, j_a < a$ and $\alpha$ is strictly increasing. Note that the size $t$ is at most $2^k$, because $q/m \le 2^k$ by (4). Figure 4 illustrates how a particular $I$ and $J$ code a circuit in the case $m = 5$ and $t = 5$ (so $q = mt = 25$). In this figure, $seq(P)$ is shown for reference, $*$ is a "don't care" symbol, and words are broken into blocks of length $m = 5$ for readability. Consider, for example, gate 6. The inputs to this gate are the inputs 3 and 4 (really, the projection functions $\xi_3$ and $\xi_4$). Because $\alpha^{-1}(6) = 10$, this information should be encoded in $m\text{-}word(I, 10)$ and $m\text{-}word(J, 10)$. Thus, we take $m\text{-}word(I, 10) = 0^3 1^2 = 00011$ and $m\text{-}word(J, 10) = 0^4 1^1 = 00001$. Consider now gate 8. The inputs to this gate are gates 5 and 7. Because $\alpha^{-1}(8) = 20$, this information should be encoded in $m\text{-}word(I, 20)$ and $m\text{-}word(J, 20)$. Thus, we take $m\text{-}word(I, 20) = 00000$ because $m\text{-}word(P, \alpha^{-1}(5)) = m\text{-}word(P, 5) = 00000$. Similarly, $m\text{-}word(J, 20) = 01000$ because $m\text{-}word(P, \alpha^{-1}(7)) = m\text{-}word(P, 15) = 01000$.

Let $U$ be a circuit of size $t = q/m$ with $m$ inputs, let $x \in \{0,1\}^m$ and $D \subseteq \mathbb{N}$. Let $\xi_i(x)$ denote the function associated with step $\beta_i$ of $q\text{-}circuit(I, J)$ for $0 \le i < m + t - 1$. Then $D$ *computes* $U$ *on* $x$ iff

$$(\forall a \in S \cup B \text{ with } a \le q)[\, a \in D \;\Leftrightarrow\; \xi_{\alpha(a)}(x) = 1\,].$$

Note in particular that $m\text{-}word(D, 0) = x$, because for $0 \le a < m$ we have $\xi_{\alpha(a)}(x) = \xi_a(x) = x_a$ where $x = x_0 x_1 \cdots x_{m-1}$. Figure 4 shows $seq(D)$ for a set $D$ that computes $q\text{-}circuit(I, J)$ on $x = 11001$. In particular, 11001 is a prefix of $seq(D)$. For example, $25 \notin D$ because $\xi_{\alpha(25)}(x) = \xi_9(x) = 0$. The latter is consistent with $D \cap \{0, 1, 2, \ldots, 24\}$ because the inputs to gate 9 are gates 8 and 6, $\alpha^{-1}(8) = 20 \in D$, $\alpha^{-1}(6) = 10 \in D$, and $g_{\mathrm{NA}}(1, 1) = 0$.

We note some simple facts and then write $\psi_5$. Claim 4.5 is immediate from definitions and the fact that $P$ is constrained as in Figure 2.




$$\text{q-circuit(I,J)} = (0, 1, g_{NA}),\ (3, 4, g_{NA}),\ (2, 6, g_{NA}),\ (5, 7, g_{NA}),\ (8, 6, g_{NA})$$
$$= \beta_5\ ,\ \beta_6\ ,\ \beta_7\ ,\ \beta_8\ ,\ \beta_9$$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| seq(P) | = | 11111 | 00000 | 10000 | 01000 | 11000 | 00100 | ... |
| seq(I) | = | ***** | 11111 | 00011 | 00111 | 00000 | 11000 | ... |
| seq(J) | = | ***** | 01111 | 00001 | 10000 | 01000 | 10000 | ... |
| seq(D) | = | 11001 | 0**** | 1**** | 1**** | 1**** | 0**** | ... |
| a | | 01234 | 5 | 10 | 15 | 20 | 25 | |
| $\propto$(a) | | 01234 | 5 | 6 | 7 | 8 | 9 | |

Figure 4: $I$ and $J$ code a circuit, *q-circuit*$(I, J)$. $D$ computes *q-circuit*$(I, J)$ on input $x = 11001$.

**Claim 4.5** (a) *Let* $x \in \{0,1\}^m$, $F \subseteq \mathbb{N}$, *and* $e = m(int(x) + 1)$. *Then* $m$-*word*$(P, e) = x$, *and* $e \in F$ *iff* $fct(F)(x) = 1$.

(b) *If* $U$ *is a circuit of size* $q/m$ *with* $m$ *inputs,* $x \in \{0,1\}^m$, *and* $D$ *computes* $U$ *on* $x$, *then* $q \in D$ *iff* $U(x) = 1$.

Now assuming that $B, P, d, q$ are as in (4), $\psi_5(F, B, P, q)$ is true iff there is a NAND-circuit of size $q/m$ that computes $fct(F)$.

$\psi_5(F, B, P, q)$ is

$$\exists I\, \exists J\, \forall e\, \exists D\, \forall a\, \exists i\, \exists j\, \psi_5' \qquad (\psi_5)$$

where $\psi'_5$ is

$$\begin{aligned}
(\, e \in B \;\Rightarrow\; \{&\textsc{Match}(D, 0, P, e) \\
&\wedge\; (a \in B \;\Rightarrow \\
&\qquad \{\textsc{Match}(P, i, I, a) \;\wedge\; \textsc{Match}(P, j, J, a) \\
&\qquad\quad \wedge\; (a \in D \;\Leftrightarrow\; \{i \notin D \;\vee\; j \notin D\})\}) \\
&\wedge\; (q \in D \;\Leftrightarrow\; e \in F)\}\,).
\end{aligned} \tag{$\psi'_5$}$$

In words, $\psi_5$ expresses the following. There exists a circuit, *q-circuit*$(I, J)$, of size $q/m$ ($\leq 2^k$) such that for all inputs $x \in \{0,1\}^m$ (where $e = enc(x) = m(int(x)+1)$) there exists a computation $D$ such that:

1. $x$ is the input to the computation $D$, that is, *m-word*$(D, 0) =$ *m-word*$(P, e)$ by Claim 4.4(a) and Claim 4.5(a).

2. For all gates $\beta_{\alpha(a)}$ with $a \in B$, there exists $i$ and $j$ such that the output $\xi_{\alpha(a)}(x)$ of $\beta_{\alpha(a)}$ is computed correctly as $(\neg\xi_{\alpha(i)}(x) \vee \neg\xi_{\alpha(j)}(x))$ (which is equivalent to $g_{\mathrm{NA}}(\xi_{\alpha(i)}(x), \xi_{\alpha(j)}(x))$). Note also by Claim 4.4(b) and (c) that when $a \in B$ and $a \leq q$, there is at most one $i$ such that $\textsc{Match}(P, i, I, a)$ is true, and this $i$ must be a "proper" value, that is, $i < a$ and either $i < m$ or $i \in B$, and similarly for $\textsc{Match}(P, j, J, a)$.

3. *q-circuit*$(I, J)(x) = 1$ iff *fct*$(F)(x) = 1$ by Claim 4.5.

Now let $\textsc{Easy}'(F)$ be

$$\exists B\, \exists P\, \exists d\, \exists q\, \exists I\, \exists J\, \forall e\, \exists D\, \forall a\, \exists i\, \exists j\, (\psi_1 \;\wedge\; \psi_2 \;\wedge\; \psi_3 \;\wedge\; \psi_4 \;\wedge\; \psi'_5).$$

Note that each of $\psi_1$, $\psi_2$, and $\psi_4$ is a conjunction of subformulas and that $\psi_3$ and $\psi'_5$ have the form $(\ldots)$. By standard manipulation of quantifiers, and using that the variables $I, J, e, D, i, j$ do not appear free in any of $\psi_1, \psi_2, \psi_3$, or $\psi_4$, the formula $\textsc{Easy}'(F)$ is equivalent to

$$\exists B\, \exists P\, \exists d\, \exists q\, (\forall a\, (\psi_1 \;\wedge\; \psi_2 \;\wedge\; \psi_3 \;\wedge\; \psi_4) \;\wedge\; \psi_5).$$

It should now be clear that $\textsc{Easy}'(F)$ is true iff there is a NAND-circuit of size $2^k$ that computes *fct*$(F)$. In the "if" direction, always choose $d = m2^m$ and $q = m2^k$, choose $P$ and $B$ as in (4), and choose legal $(I, J)$ such that *q-circuit*$(I, J)$ computes *fct*$(F)$. Note that $d = m2^m$ means that there is a one-to-one correspondence between $\{0,1\}^m$ and $B$ given by $x \leftrightarrow enc(x)$. Given $e \in B$, choose finite $D$ such that $D$ computes *q-circuit*$(I, J)$ on $x$ where $e = enc(x)$. Moreover, choose $I, J, D$ such that, for all $a$ with $a \in B$ and $a > q$,

$$\exists i\, \exists j\, (\textsc{Match}(P, i, I, a) \;\wedge\; \textsc{Match}(P, j, J, a) \;\wedge\; (a \in D \;\Leftrightarrow\; (i \notin D \;\vee\; j \notin D))).$$

(This can be done, for example, by taking *m-word*$(I, a) =$ *m-word*$(J, a) = 0^m$, and $a \in D$ iff $m \notin D$, for all $a \in B$ with $a > q$.) In the "only if" direction, $B, P, d$, and $q$ must be chosen to satisfy (4); then the choice of $I$ and $J$ determines a circuit (*q-circuit*$(I, J)$) of size $q/m \leq 2^k$ that computes *fct*$(F)$. If $q/m < 2^k$, then *q-circuit*$(I, J)$ can be easily modified to give a circuit of size $2^k$ that computes *fct*$(F)$.

We now count the length of $\textsc{Easy}'(F)$. Let $z = \lfloor \log_{10} m \rfloor + 1$. Note that $|dec(k)| = |dec(m)| = z$. First, $|\textsc{Match}| = 72 + 3z$. The lengths of $\psi_1$, $\psi_2$, $\psi_3$, $\psi_4$, and $\psi_5'$ are, respectively, $40 + 3z$, $61 + 2z$, $14 + z$, $18 + z$, and $38 + 3|\textsc{Match}|$. The length of $\textsc{Easy}'$ is the sum of these plus 28 additional symbols, so $|\textsc{Easy}'| = 199 + 7z + 3|\textsc{Match}|$.

Using the abbreviation trick with $r = 3$ and $p = 4$ to reduce the three occurrences of $\textsc{Match}$ to one, $\textsc{Easy}'$ can be written equivalently as $\textsc{Easy}$ where

$$|\textsc{Easy}| = |\textsc{Easy}'| - 2|\textsc{Match}| + 96 = 377 + 10\lfloor \log_{10} m \rfloor.$$

Note that the additional variables $d_1, d_2, d_3, d_4, y_1, y_2, y_3, y$ used in the abbreviation trick can be named $A, c, C, f, g, h, k, l$, respectively.

This completes the construction of $\textsc{Easy}(F)$ and the proof of Lemma 4.3. ■

**Lemma 4.6** *Let $k$, $m$, and $n$ be positive integers that satisfy* (1) *and* (3) *of Theorem 4.1. Then there is a function $f_0 : \{0,1\}^m \to \{0,1\}$ such that:*

(i) *$C(f_0) > 2^k/5$;*

(ii) *for each $x \in \{0,1\}^m$ there is a sentence $\varphi_x \in \mathcal{L}$ such that $|\varphi_x| = n$, and $\varphi_x \in$ EWS1S iff $f_0(x) = 1$; and*

(iii) *if $h : \Gamma \to \{0,1\}^6$ is any encoding for $\Gamma$ and if $\tau$ is the function that maps $x$ to $\hat{h}(\varphi_x)$ for all $x \in \{0,1\}^m$, then $C(\tau) \le 2^{20}m^2$.*

*Proof.* Let $k$, $m$, and $n$ be fixed positive integers that satisfy constraints (1) and (3) of Theorem 4.1. Let $\omega(x)$ be a decimal representation of $enc(x) = m(int(x)+1)$, where leading zeroes are appended if necessary so that

$$|\,\omega(x)| = \lfloor (\log_{10} 2)m \rfloor + \lfloor \log_{10} m \rfloor + 2. \tag{12}$$

Note that $x \in \{0,1\}^m$ implies $int(x) \le 2^m - 1$. So $enc(x) \le m2^m$, and the decimal representation of $enc(x)$ need never be longer than $\lfloor \log_{10}(m2^m) \rfloor + 1 \le \lfloor (\log_{10} 2)m \rfloor + \lfloor \log_{10} m \rfloor + 2$.

Define the formula $\textsc{LessThan}(G, F)$ by

$$\exists a\,(a \in F \ \wedge\ a \notin G \ \wedge\ \forall b\,(b > a \ \Rightarrow\ (b \in G \ \Leftrightarrow\ b \in F))).$$

$\textsc{LessThan}(G, F)$ is easily seen to define a linear order on finite subsets of $\mathbb{N}$. Also, for each finite $X \subset \mathbb{N}$ there is a finite number of $W \subset \mathbb{N}$ satisfying $\textsc{LessThan}(W, X)$.

Now let $\varphi_x'''$ be the following sentence:

$$\exists F\,\forall G\,(\omega(x) \in F \ \wedge\ \neg\textsc{Easy}(F) \ \wedge\ (\textsc{LessThan}(G, F) \ \Rightarrow\ \textsc{Easy}(G))). \tag{$\varphi_x'''$}$$

A particular NAND-circuit of size $2^k$ is completely described by giving a pair $(i, j)$ with $0 \le i, j < s$ for each step $s$ with $m \le s \le m + 2^k - 1$. Therefore, the number of NAND-circuits of size $2^k$ is at most $(m + 2^k)^{2 \cdot 2^k} < 2^{2^m}$, where the inequality follows from constraint (1) of Theorem 4.1. Because there are $2^{2^m}$ functions from $\{0,1\}^m$ to $\{0,1\}$, there is some finite $X \subset \mathbb{N}$ such that $\textsc{Easy}(X)$ is false.

Because there is a finite number of $W \subset \mathbb{N}$ with $\textsc{LessThan}(W, X)$, there is exactly one $F_0 \subset \mathbb{N}$ such that $\forall G\,(\neg\textsc{Easy}(F_0) \ \wedge\ (\textsc{LessThan}(G, F_0) \ \Rightarrow\ \textsc{Easy}(G)))$ is true. We take $f_0 = fct(F_0)$.

Because each function in $\Omega_{16}$ can be synthesized using at most five NAND-gates [34], it follows that $C(f_0) > 2^k/5$. Also, "$\omega(x) \in F$" is true iff $enc(x) \in F_0$ iff $f_0(x) = 1$ by definition of $f_0 = fct(F_0)$, so $\varphi'''_x$ is true iff $f_0(x) = 1$.

Substituting the definition of LESSTHAN in the definition of $\varphi'''_x$, it can be written equivalently as $\varphi''_x$, defined by

$$\begin{aligned}&\exists F\,\forall G\,\forall a\,\exists b\,(\omega(x) \in F \;\wedge\; \neg\textsc{Easy}(F)\\ &\quad\wedge\; (\{a \notin F \;\vee\; a \in G \;\vee\; \{b > a \;\wedge\; (b \notin G \;\Leftrightarrow\; b \in F)\}\} \;\vee\; \textsc{Easy}(G)\,)).\end{aligned} \qquad (\varphi''_x)$$

To see the equivalence, let $\theta(a, b, G, F)$ denote the formula within the outermost $\{\ldots\}$, and note that $\forall a\,\exists b\,(\theta(a, b, G, F))$ is equivalent to $\neg\textsc{LessThan}(G, F)$. Using (12),

$$|\varphi''_x| = 41 + 2|\textsc{Easy}| + \lfloor(\log_{10}2)m\rfloor + \lfloor\log_{10}m\rfloor.$$

The abbreviation trick with $r = 2$ and $p = 1$ applied to $\varphi''_x$ and EASY gives $\varphi'_x$ equivalent to $\varphi''_x$ and

$$|\varphi'_x| = |\varphi''_x| - |\textsc{Easy}| + 41 = 459 + \lfloor(\log_{10}2)m\rfloor + 11\lfloor\log_{10}m\rfloor$$

using Lemma 4.3(b) for an upper bound on $|\textsc{Easy}|$. The additional variables $d_1, y_1, y_2, y$ can be named $E, m, n, o$, respectively. By constraint (3) of Theorem 4.1, $j \geq 0$ can be chosen such that $\varphi_x = \varphi'_x\#^j$ and $|\varphi_x| = n$. So $f_0$ and $\varphi_x$ satisfy the requirements of Lemma 4.6.

It remains only to bound the circuit complexity of the transformation $\tau$ mapping $x$ to $\hat{h}(\varphi_x)$. For fixed $k$, $m$, and $n$, the word $\hat{h}(\omega(x))$ is the only part of $\hat{h}(\varphi_x)$ that depends on $x$. Recall that the length of $\omega(x)$ is independent of $x$. Thus, all bits of $\hat{h}(\varphi_x)$, excluding $\hat{h}(\omega(x))$, can be computed using the two gates with constant output. Now $2^{20}m^2 - 2$ is a gross upper bound on the circuit complexity of the function mapping $x$ to $\hat{h}(\omega(x))$, using straightforward classical algorithms for binary addition, binary multiplication, and binary-to-decimal conversion (see, for example, [42]).

This completes the proof of Lemma 4.6. ■

*Proof of Theorem 4.1.* Let $k$, $m$, and $n$ satisfy the constraints (1), (2), and (3) of the theorem. Assume the conclusion is false, that is, $C_{\mathrm{EWS1S}}(n) \leq 2^{k-3}$. Therefore, there is an encoding $h : \Gamma \to \{0, 1\}^6$ and a circuit $U$ of size $\leq 2^{k-3}$ with $6n$ inputs that computes a function $f$ such that $f(\hat{h}(\varphi)) = 1$ iff $\varphi \in \mathrm{EWS1S}$, for all $\varphi \in \mathcal{L} \cap \Gamma^n$.

Let $f_0$ and $\tau$ be as in Lemma 4.6 for this $k, m, n$, and $h$. In particular, $C(f_0) > 2^k/5$. Let $T$ be a circuit of size $\leq 2^{20}m^2$ that computes $\tau$. Let $U_0$ be the circuit obtained by identifying the $6n$ outputs of $T$ with the $6n$ inputs of $U$. Thus, the output $\hat{h}(\varphi_x)$ of $T$ becomes the input of $U$. It is clear how to define $U_0$ from $T$ and $U$ within the formalism of straightline algorithms such that the size of $U_0$ is the size of $T$ plus the size of $U$. Now $U_0$ computes $f_0$ because, for all $x \in \{0, 1\}^m$,

$$U_0(x) = 1 \;\Leftrightarrow\; f(\hat{h}(\varphi_x)) = 1 \;\Leftrightarrow\; \varphi_x \in \mathrm{EWS1S} \;\Leftrightarrow\; f_0(x) = 1.$$

Because constraint (2) implies $2^{20}m^2 \leq 2^{k-4}$, the size of $U_0$ is at most $2^{k-4} + 2^{k-3} < 2^k/5$. This contradicts $C(f_0) > 2^k/5$, so we must have $C_{\mathrm{EWS1S}}(n) > 2^{k-3}$.

**Remark 4.7 (What fraction of inputs are hard?)** Theorem 4.2, read literally, says that for each circuit of size $10^{125}$ there is an input of length 610 on which the circuit gives the wrong answer about membership of the input in EWS1S. By our proof, we may assume that this input is a sentence of EWS1S, because $\tau$ produces only (encodings of) sentences. Some improvement in the *number* of hard sentences can be made, using the simple argument that if a circuit is wrong for only $w$ sentences then the circuit can be patched by building in a table of the answers for these sentences, increasing the size of the circuit by at most $6nw + 2$. For example, if a circuit is wrong on at most $10^{121}$ sentences of length 610, then its size must be at least $10^{124}$. Although large in absolute terms, $10^{121}$ is an insignificant fraction of the number of sentences of length 610. When this argument is used for arbitrary $n$, the provably hard sentences form an asymptotically vanishing fraction. Except for this most trivial argument, we do not know how to obtain lower bounds on the frequency of hard inputs. This question is interesting, important, and wide open. We do not even know whether some constant fraction of the sentences must be hard. To be specific, is there a constant $\varepsilon > 0$, such that for all polynomials $p(n)$, there exists an $n_0$, such that for all $n \geq n_0$, all circuits $U_n$ of size $p(n)$, and all encodings $h$, the circuit $U_n$ decides EWS1S incorrectly ($U_n(\hat{h}(\varphi)) = 1$ iff $\varphi \notin$ EWS1S) on at least an $\varepsilon$ fraction of the sentences $\varphi$ of length $n$?

Two developments related to the frequency of hard inputs should be mentioned. First, Levin [45] (see also Gurevich [32]) has introduced a theory of average-case complexity. Here a decision problem has the form $(L, \mu)$ where $L \subseteq \Gamma^*$ and $\mu$ is a probability distribution on $\Gamma^*$. Part of the theory is a notion of efficient reduction that is "distribution preserving." Second, Ajtai [6] has shown a connection between the worst-case and average-case complexities of the problem of finding a shortest non-zero vector in an integer lattice.

### 4.3 Extension to probabilistic circuits

Beginning with Gill [31], Rabin [59], and Solovay and Strassen [72], solution of decision problems by probabilistic algorithms, that is, algorithms employing random numbers, has played an increasing role in the theory of computing. Besides being a natural mathematical concept, probabilistic computation can be useful in practice: if the error probability of a probabilistic decision algorithm is bounded below 1/2 then the error probability can be decreased to $2^{-k}$ by making $O(k)$ independent runs of the algorithm and taking the majority answer (this follows from Fact 4.8 below). It has even been suggested that the current theoretical model of the "tractable problems," namely P, should be replaced by the class BPP [31] containing the languages accepted by polynomial-time probabilistic Turing machines with error probability bounded below 1/2.

The definition of a *probabilistic circuit of size $t$ with $m$ inputs* is like the definition of a circuit in Section 2.1, with one exception. In addition to the $m$ inputs that receive an input $x \in \{0,1\}^m$, there is some number $l$ of *random bit inputs*. Formally, these are added as functions $\xi_k$ for $-l \leq k \leq -1$, the projection functions of the random bit inputs. Now each function $\xi_k$, for $-l \leq k \leq m+t-1$, maps $\{0,1\}^l \times \{0,1\}^m$ to $\{0,1\}$ in the obvious way. For such a circuit $U$, for $r \in \{0,1\}^l$ and $x \in \{0,1\}^m$, let $U(r,x) = \xi_{m+t-1}(r,x)$, the associated function of the highest numbered step. Let $p_U(x)$ be the probability that $U(r,x) = 1$ when $r$ is chosen uniformly at random from $\{0,1\}^l$.

Let $f : \{0,1\}^m \to \{0,1\}$, and let $0 < \varepsilon < 1/2$. The *$\varepsilon$-error probabilistic circuit complexity of $f$*, which we denote $PC_\varepsilon(f)$, is the smallest $t$ such that there is a probabilistic circuit of size $t$ with $m$ inputs such that, for all $x \in \{0,1\}^m$, if $f(x) = 1$ then $p_U(x) \geq 1 - \varepsilon$, and if $f(x) = 0$ then $p_U(x) \leq \varepsilon$. Now for $L \subseteq \Gamma^*$, the definition of the *$\varepsilon$-error probabilistic circuit complexity of $L$*, denoted $PC_{L,\varepsilon}(n)$, is analogous to the definition of $C_L(n)$ in Section 2.1: it is the minimum of

$PC_\varepsilon(f)$ over $f \in \mathcal{F}_{L,n}$.

Bennett and Gill [11] have shown that for each $\Gamma$ and $\varepsilon$ there is a constant $c$ such that for all $L \subseteq \Gamma^*$, $C_L(n) \leq cn \cdot PC_{L,\varepsilon}(n)$.[10] In particular, P/poly = BPP/poly; *i.e.*, polynomial-size probabilistic circuits are no more powerful than polynomial-size deterministic circuits. To prove a numerical lower bound on $PC_{\mathrm{EWS1S},\varepsilon}(n)$, however, we need a value for $c$. To obtain this, we use the following Chernoff bound [51, Chap. 4].

**Fact 4.8** *Let $X_1, \ldots, X_N$ be independent 0/1 valued random variables and let $0 < \varepsilon < 1$ be such that* $\Pr[X_i = 1] = \varepsilon$ *for $1 \leq i \leq N$. For all $\delta > 0$,*

$$\Pr[\, X_1 + \cdots + X_N > (1+\delta)\varepsilon N \,] < \left( \frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^{\varepsilon N}. \tag{13}$$

The following uses the same proof method as [1, 11], but includes values for constants. Let $\ln x = \log_e x$.

**Theorem 4.9** *Let $\Gamma$ be a finite alphabet, $L \subseteq \Gamma^*$, and $0 < \varepsilon < 1/2$. Let $\delta = 1/(2\varepsilon) - 1$ and $c = (\varepsilon((1+\delta)\ln(1+\delta) - \delta))^{-1}$. Then*

$$C_L(n) \leq (\lceil cn \ln|\Gamma| \rceil + 1)(PC_{L,\varepsilon}(n) + 14). \tag{14}$$

*Proof.* Let $s = \lceil \log|\Gamma| \rceil$. Fix an $n$, and restrict attention to those $x \in \Gamma^n$. Let $U$ be a probabilistic circuit of size $PC_{L,\varepsilon}(n)$ with inputs $x' \in \{0,1\}^{sn}$ and $r \in \{0,1\}^l$, and let $h$ be an encoding for $\Gamma$ such that, for all $x$ (in $\Gamma^n$), if $x \in L$ then $p_U(\hat{h}(x)) \geq 1 - \varepsilon$, and if $x \notin L$ then $p_U(\hat{h}(x)) \leq \varepsilon$. For odd $N \in \mathbb{N}$, we define a probabilistic circuit $U_N$. The circuit $U_N$ contains $N$ copies of $U$. The input $\hat{h}(x)$ is given to all copies, but the random bits are chosen independently for each copy. Thus, $U_N$ has inputs $r', x'$ where $r' \in \{0,1\}^{Nl}$ and $x' \in \{0,1\}^{sn}$. The outputs of the copies are fed into a circuit that computes $maj_N$, the $N$-ary majority function ($maj_N(b_1, \ldots, b_N) = 1$ iff $\sum_{i=1}^N b_i \geq N/2$). Thus, $U_N(r', x') = 1$ iff a majority of the $N$ copies of $U$ output 1. Because $N$ is odd, a majority is always a strict majority. Using the rough bound $C(maj_N) \leq 14N$ for all $N$ (which follows easily from [65, Thm. 3.1.1.4]), the size of $U_N$ is at most $N(PC_{L,\varepsilon}(n) + 14)$.

Define the *error probability of* $U_N$, denoted $err(N)$, to be the smallest number $\gamma$ such that, for all $x$, if $x \in L$ then $p_{U_N}(\hat{h}(x)) \geq 1 - \gamma$, and if $x \notin L$ then $p_{U_N}(\hat{h}(x)) \leq \gamma$. Clearly

$$err(N) \leq \Pr[\, X_1 + \cdots + X_N > N/2 \,] \quad \text{where} \quad \Pr[X_i = 1] = \varepsilon \text{ for } 1 \leq i \leq N$$

and where the $X_i$ are independent. By Fact 4.8, $err(N)$ is less than the RHS of (13) where $\delta = 1/(2\varepsilon) - 1$ is chosen so that $(1+\delta)\varepsilon = 1/2$. Thus, $err(N) < \alpha^{-N}$, where $\alpha = ((1+\delta)^{(1+\delta)} e^{-\delta})^\varepsilon$. Note that $c = (\ln \alpha)^{-1}$. We want to choose $N$ large enough that $err(N) < |\Gamma|^{-n}$. This holds if $\alpha^{-N} \leq |\Gamma|^{-n}$, which is equivalent to $N \geq (\ln \alpha)^{-1} n \ln|\Gamma|$. So there is an odd $N \leq \lceil cn \ln|\Gamma| \rceil + 1$ such that $err(N) < |\Gamma|^{-n}$. Thus, the size of $U_N$ is at most the RHS of (14).

Define $\chi_L : \Gamma^n \to \{0,1\}$ by $\chi_L(x) = 1$ if $x \in L$, or 0 if $x \notin L$. To finish the proof, we show that if $err(N) < |\Gamma|^{-n}$ then $(\exists\, r')(\forall x)[U_N(r', \hat{h}(x)) = \chi_L(x)]$. Thus, by substituting some string

[10] Adleman [1], using a similar proof, had earlier shown this for probabilistic circuits with "one-sided error," *i.e.*, there exists $\varepsilon > 0$ such that, for all $x$, if $f(x) = 1$ then $p_U(x) \geq \varepsilon$, and if $f(x) = 0$ then $p_U(x) = 0$.

$r_0' \in \{0,1\}^{Nl}$ for the random inputs of $U_N$, we obtain a (deterministic) circuit $D$ that computes $\chi_L$. The size of $D$ is at most the size of $U_N$, because gates of $U_N$ that use a random bit as input can be simplified to obtain $D$. Assume for contradiction that $(\forall r')(\exists x)[U_N(r', \hat{h}(x)) \neq \chi_L(x)]$. For $r' \in \{0,1\}^{Nl}$ and $x \in \Gamma^n$, let $W(r', x) = U_N(r', \hat{h}(x)) \oplus \chi_L(x)$ (intuitively, $W(r', x) = 1$ iff $U_N$ is wrong on input $x$ and random string $r'$). By assumption, $(\forall r')(\exists x)[W(r', x) = 1]$, so $\sum_{r'} \sum_x W(r', x) \geq 2^{Nl}$. But $err(N) < |\Gamma|^{-n}$ implies for each $x$ that $\sum_{r'} W(r', x) < |\Gamma|^{-n} 2^{Nl}$, which implies $\sum_x \sum_{r'} W(r', x) < 2^{Nl}$. This contradiction proves the lemma. ■

To state a version of Theorem 1.1 for probabilistic circuit complexity, we take $\varepsilon = 1/3$ to be definite.

**Theorem 4.10** $PC_{\mathrm{EWS1S},1/3}(614) > 10^{125}$.

*Proof.* Taking $k = 435$, $m = 445$, and $n = 614$ in Theorem 4.1 gives $C_{\mathrm{EWS1S}}(614) > 2^{432}$. Recall that $|\Gamma| = 63$. Looking now at the statement of Theorem 4.9, a calculation shows that $c \ln 63 < 115$ when $\varepsilon = 1/3$. Theorem 4.9 gives

$$PC_{\mathrm{EWS1S},1/3}(614) \geq \frac{C_{\mathrm{EWS1S}}(614)}{115 \cdot 614 + 1} - 14 > \frac{2^{432}}{10^5} > 10^{125}.$$

■

## 4.4 Quantum circuit complexity

An active area during the past ten years has been *quantum computational complexity*, where algorithms are performed by quantum-mechanical systems. Introductions to quantum computation include the book by Nielsen and Chuang [52], the survey by Aharonov [3], and the collection of papers in the October 1997 issue of *SIAM Journal on Computing*. Quantum circuits were first defined by Deutsch [24] and study of quantum circuit complexity was initiated by Yao [82]. A definition of quantum circuits is beyond the scope of this paper.

We do not have a nontrivial numerical lower bound on the quantum circuit complexity of EWS1S. A major roadblock is that an analogue of Theorem 4.9 for quantum circuits is not known. The quantum circuit complexity of a problem is linear in its deterministic circuit complexity, but it is possible, given current knowledge, that deterministic circuit complexity can be exponentially larger than quantum circuit complexity. Absent an analogue of Theorem 4.9, it seems that a diagonalization-based proof of a large lower bound on quantum circuit complexity must start from scratch, for example, use a formula in EWS1S (or some other decidable theory) to diagonalize over quantum circuits. We see no serious technical obstacles to doing this, perhaps using an input length in the thousands rather than the hundreds, although we have not tried.

There is, however, an analogue of Theorem 3.1: There is a language in ESPACE whose quantum circuit complexity is exponential a.e. The proofs are essentially the same, although two facts about quantum complexity are needed (stated here informally). First, there is a finite complete basis for quantum circuit computation, that is, a finite set of quantum gates that suffice to build any quantum circuit [2, 73]. Using this finite basis, Shannon's counting argument shows that the maximum $n$-ary quantum circuit complexity is exponential in $n$. The second fact is that the analogue of $p_U(x)$ for quantum circuits can be computed by a Turing machine using space polynomial in $|U| + |x|$ [14]. Using these facts, a diagonalization as in the proof of Theorem 3.2 can be done in exponential space. A corollary is that if ESPACE is poly-lin reducible to $L$, then the quantum circuit complexity of $L$ is exponential. For example, the quantum circuit complexities of WS1S and $\mathrm{Th}\langle \mathbb{N}, + \rangle$ are exponential a.e.

## 5 Conclusion and Future Directions

We have shown that a method based on diagonalization can be used to prove an astronomical lower bound on the computational complexity of a natural, uncontrived and decidable problem in logic, even when the problem is restricted to input lengths in the hundreds. We have extended the result to probabilistic circuits. We believe that a similar result can be shown for quantum circuit complexity, perhaps using input lengths in the thousands.

Our purpose has been to demonstrate by example (EWS1S) that large lower bounds can be proved for relatively small inputs. We believe that this phenomenon is typical, and that similar results can be obtained for most of the exponential asymptotic lower bounds in the literature.

Our main result disposes of the concern that the exponential lower bound on the complexity of EWS1S begins to take effect only for impractically huge inputs. However, this result does not eliminate another possible loophole, that EWS1S might be hard only for an insignificant fraction of the sentences, as discussed in Remark 4.7. Obtaining nontrivial bounds on the density of hard inputs is an important open problem.

## References


[1] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 75–83. IEEE Computer Society Press, Los Alamitos, CA, 1978.

[2] L. M. Adleman, J. Demarrais, and M.-D. A. Huang. Quantum computability. *SIAM J. Comput.*, 26:1524–1540, 1997.

[3] D. Aharonov. Quantum computation – a review. In D. Stauffer, editor, *Annual Reviews of Computational Physics, Vol. IV*. World Scientific Press, River Edge, NJ, 1998.

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[5] M. Ajtai. $\Sigma_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[6] M. Ajtai. Generating hard instances of lattice problems. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. ACM, New York, 1996. Full version: Report TR96-007, *Electronic Colloquium on Computational Complexity*, http://www.eccc.uni-trier.de/eccc/.

[7] E. Allender. Some pointed questions concerning asymptotic lower bounds and news from the isomorphism front. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, pages 25–41. World Scientific Press, River Edge, NJ, 2001.

[8] N. Alon and R. B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7:1–23, 1987.

[9] A. E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Dokl. Ak. Nauk SSSR*, 282:1033–1037, 1985. English translation in *Sov. Math. Dokl. 31*, 530–534.

[10] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4:431–442, 1975.

[11] C. H. Bennett and J. Gill. Relative to a random oracle $A$, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. *SIAM J. Comput.*, 10:96–113, 1981.

[12] L. Berman. The complexity of logical theories. *Theoret. Comput. Sci.*, 11:71–77, 1980.

[13] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM J. Comput.*, 6:305–322, 1977.

[14] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26:1411–1473, 1997.

[15] M. Blum. Recursive function theory and speed of computation. *Canadian Math. Bull.*, 9:745–749, 1966.

[16] N. Blum. A Boolean function requiring $3n$ network size. *Theoret. Comp. Sci.*, 28:337–345, 1984.

[17] R. B. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A, Algorithms and Complexity*, chapter 14, pages 757–804. Elsevier, New York, 1990.

[18] A. Borodin. On relating time and space to size and depth. *SIAM J. Comput.*, 6:733–743, 1977.

[19] J. R. Büchi. Weak second order arithmetic and finite automata. *Z. Math. Logik Grundl. Math.*, 6:66–92, 1960.

[20] G. J. Chaitin. The Berry paradox. *Complexity*, 1:26–30, 1995.

[21] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. Assoc. Comput. Mach.*, 28:114–133, 1981.

[22] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the 1964 International Congress for Logic, Methodology and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1965.

[23] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, New York, 1971.

[24] D. Deutsch. Quantum computational networks. *Proc. Roy. Soc. Lond. A*, 425:73–90, 1989.

[25] P. E. Dunne. *The Complexity of Boolean Networks.* Academic Press, New York, 1988.

[26] J. Edmonds. Paths, trees and flowers. *Canadian J. Math.*, 17:449–467, 1965.

[27] A. Ehrenfeucht. Practical decidability. *J. Comput. Syst. Sci.*, 11:392–396, 1975.

[28] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–51, 1961.

[29] M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburger arithmetic. In R. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, vol. 7*, pages 27–42. American Mathematical Society, Providence, RI, 1974.

[30] M. Furst, M. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. *Math. Syst. Th.*, 17:13–27, 1984.

[31] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6:675–695, 1977.

[32] Y. Gurevich. Average case completeness. *J. Comput. Syst. Sci.*, 42:346–398, 1991.

[33] L. H. Harper, W. N. Hsieh, and J. E. Savage. A class of Boolean function with linear combinational complexity. *Theoret. Comp. Sci.*, 1:161–183, 1975.

[34] M. A. Harrison. *Introduction to Switching and Automata Theory.* McGraw-Hill, New York, 1965.

[35] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117:285–306, 1965.

[36] J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Advances in Computer Research, Vol. 5: Randomness and Computation.* JAI Press, Greenwich, CT, 1986.

[37] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[38] R. Impagliazzo and A. Wigderson. P=BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229. ACM, New York, 1997.

[39] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Inform. and Control*, 55:40–56, 1982.

[40] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[41] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 302–309. ACM, New York, 1980.

[42] D. E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms.* Addison-Wesley, Reading, MA, 1969.

[43] D. E. Knuth. Mathematics and computer science: coping with finiteness. *Science*, 194:1235–1242, 1976. Reprinted in: D. E. Knuth, *Selected Papers in Computer Science*, Cambridge University Press, Cambridge, UK, 1996.

[44] L. A. Levin. Universal sorting problems. *Problemy Peredaci Informacii*, 9:115–116, 1973. English translation in *Problems of Information Transmission 9*, 265–266.

[45] L. A. Levin. Average case complete problems. *SIAM J. Comput.*, 15:285–286, 1986.

[46] M. Li and P. M.B. Vitányi. Kolmogorov complexity and its applications. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A, Algorithms and Complexity*, chapter 4, pages 187–254. Elsevier, New York, 1990.

[47] O. B. Lupanov. Ob odnom metode sinteza skhem. *Izv. VUZ (Radiofizika)*, 1:120–140, 1958.

[48] A. R. Meyer. Weak monadic second-order theory of successor is not elementary-recursive. In R. Parikh, editor, *Logic Colloquium: Symposium on Logic Held at Boston 1972-73, Lecture Notes in Mathematics, Vol. 453*, pages 132–154. Springer-Verlag, Berlin, 1975.

[49] A. R. Meyer and E. M. McCreight. Computationally complex and pseudo-random zero-one valued functions. In *Theory of Machines and Computations*, pages 19–42. Academic Press, New York, 1971.

[50] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society Press, Los Alamitos, CA, 1972.

[51] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, Cambridge, UK, 1995.

[52] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information Theory.* Cambridge University Press, Cambridge, UK, 2000.

[53] D. Osherson. Probability judgement. In E. E. Smith and D. Osherson, editors, *An Invitation to Cognitive Science, Vol. 3: Thinking.* MIT Press, Cambridge, MA, 2nd edition, 1995.

[54] W. Paul. A $2.5n$-lower bound on the combinational complexity of Boolean functions. *SIAM J. Comput.*, 6:427–443, 1977.

[55] N. Pippenger. On simultaneous resource bounds. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, pages 307–311. IEEE Computer Society Press, Los Alamitos, CA, 1979.

[56] N. Pippenger and M. J. Fischer. Relations among complexity measures. *J. Assoc. Comput. Mach.*, 26:361–381, 1979.

[57] F. Pohl. *Beyond the Blue Event Horizon*, chapter 12. Ballantine Books, New York, 1980.

[58] M. O. Rabin. Degree of difficulty of computing a function and a partial ordering of recursive sets. Technical Report 2, Hebrew Univ., Jerusalem, Israel, 1960.

[59] M. O. Rabin. Probabilistic algorithms. In J. F. Traub, editor, *Algorithms and Complexity, New Directions and Recent Trends*, pages 21–29. Academic Press, New York, 1976.

[60] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Ak. Nauk SSSR*, 281:798–801, 1985. English translation in *Sov. Math. Dokl. 31*, 354–357.

[61] E. L. Robertson. Structure of complexity in the weak monadic second-order theories of the natural numbers. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing*, pages 161–171. ACM, New York, 1974.

[62] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967.

[63] W. L. Ruzzo. On uniform circuit complexity. *J. Comput. Syst. Sci.*, 22:365–383, 1981.

[64] J. E. Savage. Computational work and time on finite machines. *J. Assoc. Comput. Mach.*, 19:660–674, 1972.

[65] J. E. Savage. *The Complexity of Computing.* Wiley, New York, 1976.

[66] B. Scarpellini. Complex Boolean networks obtained by diagonalization. *Theoret. Comput. Sci.*, 36:119–125, 1985.

[67] C. P. Schnorr. Zwei lineare untere shranken für die komplexität Boolescher funktionen. *Computing*, 13:155–171, 1974.

[68] C. P. Schnorr. A lower bound on the number of additions in monotone computations. *Theoret. Comp. Sci.*, 2:305–315, 1976.

[69] C. P. Schnorr. The network complexity and the Turing machine complexity of finite functions. *Acta Inform.*, 7:95–107, 1976.

[70] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell Syst. Tech. J.*, 28:59–98, 1949.

[71] L. A. Sholomov. On one sequence of functions which is hard to compute. *Mat. Zametki*, 17:957–966, 1975.

[72] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM J. Comput.*, 6:84–85, 1977. Erratum: *SIAM J. Comput. 7*, 118.

[73] R. Solovay and A. Yao. Manuscript, 1996.

[74] L. J. Stockmeyer. The complexity of decision problems in automata theory and logic. Technical Report MAC TR-133, MIT Project MAC, Cambridge, MA, 1974.

[75] L. J. Stockmeyer. On the combinational complexity of certain symmetric Boolean functions. *Math. Syst. Th.*, 10:323–366, 1977.

[76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comp. Sci.*, 3:1–22, 1977.

[77] L. J. Stockmeyer. Classifying the computational complexity of problems. *J. Symbolic Logic*, 52:1–43, 1987.

[78] L. J. Stockmeyer and A. K. Chandra. Intrinsically difficult problems. *Scientific American*, 240 (May):140–159, 1979.

[79] I. Wegener. *The Complexity of Boolean Functions.* Wiley-Teubner, New York, Stuttgart, 1987.

[80] C. B. Wilson. Relativized circuit complexity. *J. Comput. Syst. Sci.*, 31:169–181, 1985.

[81] A. C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26rd Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, Los Alamitos, CA, 1985.

[82] A. C. Yao. Quantum circuit complexity. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 352–361. IEEE Computer Society Press, Los Alamitos, CA, 1993.