THEORY OF COMPUTING, Volume 13 (4), 2017, pp. 1–22
www.theoryofcomputing.org

On the (Non) NP-Hardness of Computing Circuit Complexity

Cody D. Murray* R. Ryan Williams†

Received October 30, 2015; Revised February 15, 2017; Published June 30, 2017

Abstract: The Minimum Circuit Size Problem (MCSP) is: *given the truth table of a Boolean function f and a size parameter k, is the circuit complexity of f at most k*? This is the definitive problem of circuit synthesis, and it has been studied since the 1950s. Unlike many problems of its kind, MCSP is *not* known to be NP-hard, yet an efficient algorithm for this problem also seems very unlikely: for example, MCSP $\in$ P would imply there are no pseudorandom functions.

Although most NP-complete problems are complete under strong "local" reduction notions such as polylogarithmic-time projections, we show that MCSP is *provably not* NP-hard under $O(n^{1/2-\varepsilon})$-time projections, for every $\varepsilon > 0$, and is not NP-hard under randomized $O(n^{1/5-\varepsilon})$-time projections, for every $\varepsilon > 0$. We prove that the NP-hardness of MCSP under (logtime-uniform) AC0 reductions would imply extremely strong lower bounds: $\mathsf{NP} \not\subset \mathsf{P/poly}$ and $\mathsf{E} \not\subset$ i.o.-$\mathsf{SIZE}(2^{\delta n})$ for some $\delta > 0$ (hence $\mathsf{P} = \mathsf{BPP}$ also follows). We

A preliminary version of this paper appeared in the Proceedings of the 30th IEEE Conference on Computational Complexity, 2015 [21].

*Supported by NSF CCF-1212372.

†Supported in part by a David Morgenthaler II Faculty Fellowship, a Sloan Fellowship, and NSF CCF-1212372. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

ACM Classification: F.2.3, F.1.3

AMS Classification: 68Q15, 68Q17

Key words and phrases: circuit lower bounds, reductions, NP-completeness, projections, minimum circuit size problem

DOI: 10.4086/toc.2017.v013a004

show that even the NP-hardness of MCSP under general polynomial-time reductions would separate complexity classes: $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P}/\mathrm{poly}$, which implies $\mathsf{EXP} \neq \mathsf{ZPP}$. These results help explain why it has been so difficult to prove that MCSP is NP-hard.

We also consider the nondeterministic generalization of MCSP: the Nondeterministic Minimum Circuit Size Problem (NMCSP), where one wishes to compute the *nondeterministic* circuit complexity of a given function. We prove that the $\Sigma_2\mathsf{P}$-hardness of NMCSP, even under arbitrary polynomial-time reductions, would imply $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$.

1 Introduction

The Minimum Circuit Size Problem (MCSP) is the canonical logic synthesis problem: we are given $\langle T, k \rangle$ where T is a string of $n = 2^\ell$ bits (for some ℓ), k is a positive integer (encoded in binary or unary), and the goal is to determine if T is the truth table of a Boolean function with circuit complexity at most k. (For concreteness, let us say our circuits are defined over AND, OR, NOT gates of fan-in at most 2.) MCSP is in NP, because any circuit of size at most k could be guessed nondeterministically in $O(k \log k) \leq O(n)$ time, then verified on all bits of the truth table T in $\mathrm{poly}(2^\ell, k) \leq \mathrm{poly}(n)$ time.

MCSP is natural and basic, but unlike thousands of other computational problems studied over the last 40 years, the complexity of MCSP has yet to be determined. The problem could be NP-complete, it could be NP-intermediate, or it could even be in P. (It is reported that Levin delayed publishing his initial results on NP-completeness out of wanting to include a proof that MCSP is NP-complete [4]. More notes on the history of this problem can be found in [20].)

Lower bounds for MCSP**?** There is substantial evidence that $\mathrm{MCSP} \notin \mathsf{P}$. If $\mathrm{MCSP} \in \mathsf{P}$, then (essentially by definition) there are no pseudorandom functions. Kabanets and Cai [20] made this critical observation, noting that the hardness of factoring Blum integers implies that MCSP is hard. Allender et al. [5] strengthened these results considerably, showing that Discrete Log and many approximate lattice problems from cryptography are solvable in $\mathsf{BPP}^{\mathrm{MCSP}}$ and Integer Factoring is in $\mathsf{ZPP}^{\mathrm{MCSP}}$. (Intuitively, this follows from known constructions of pseudorandom functions based on the hardness of problems such as Discrete Log [13].) Allender and Das [6] recently showed that Graph Isomorphism is in $\mathsf{RP}^{\mathrm{MCSP}}$, and in fact every problem with statistical zero-knowledge interactive proofs [14] is in promise-BPP with a MCSP oracle.

NP**-hardness for** MCSP**?** These reductions indicate strongly that MCSP is not solvable in randomized polynomial time; perhaps it is NP-complete? Evidence for the NP-completeness of MCSP has been less conclusive. The variant of the problem where we are looking for a minimum size DNF (instead of an arbitrary circuit) is known to be NP-complete [11, 7, 12]. Kabanets and Cai [20] show that, if MCSP is NP-complete under so-called "natural" polynomial-time reductions (where the circuit size parameter k output by the reduction is a function of only the input length to the reduction) then $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$, and $\mathsf{E} \not\subset \mathsf{SIZE}(2^{\varepsilon n})$ for some $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{SUBEXP}$. Therefore NP-completeness under a restricted reduction type would imply (expected) circuit lower bounds. This does not necessarily show that such reductions do not *exist*, but rather that they will be difficult to construct.

Allender et al. [5] show that if $\mathsf{PH} \subset \mathsf{SIZE}(2^{n^{o(1)}})$ then (a variant of) MCSP is not hard for $\mathsf{TC0}$ under $\mathsf{AC0}$ reductions. The generalization MCSP^A for circuits with A-oracle gates has also been studied; it is known for example that $\mathrm{MCSP}^{\mathrm{QBF}}$ is complete for PSPACE under ZPP reductions [5], and recently Allender, Holden, and Kabanets [8] proved that $\mathrm{MCSP}^{\mathrm{QBF}}$ is not PSPACE-complete under logspace reductions. They also showed, among similar results, that if there is a set $A \in \mathsf{PH}$ that such that MCSP^A is hard for P under $\mathsf{AC0}$ reductions, then $\mathsf{P} \neq \mathsf{NP}$.

NP-completeness has been defined for many different reducibility notions: polynomial time, logarithmic space, $\mathsf{AC0}$, even logarithmic time reductions. In this paper, we study the possibility of MCSP being NP-complete for these reducibilities. We prove several new results in this direction, summarized as follows.

1. Under "local" polynomial-time reductions where any given output bit can be computed in $n^{o(1)}$ time, MCSP is *provably not* NP-complete, contrary to many other natural NP-complete problems. (In fact, even PARITY cannot reduce to MCSP under such reductions: see Theorem 1.3.)
2. Under slightly stronger reductions such as uniform $\mathsf{AC0}$, the NP-completeness of MCSP would imply[1] $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$ and $\mathsf{E} \not\subset$ i.o.-$\mathsf{SIZE}(2^{\delta n})$ for some $\delta > 0$, therefore $\mathsf{P} = \mathsf{BPP}$ as well by [19].
3. Under the strongest reducibility notions such as polynomial time, the NP-completeness of MCSP would still imply major separations of complexity classes. For example, $\mathsf{EXP} \neq \mathsf{ZPP}$ would follow, a major (embarrassingly) open problem.

Together, the above results tell a convincing story about why MCSP has been difficult to prove NP-complete (if that is even true). Part 1 shows that, unlike many textbook reductions for NP-hardness, no simple "gadget-based" reduction can work for proving the NP-hardness of MCSP. Part 2 shows that going only a little beyond the sophistication of textbook reductions would separate P from NP and fully derandomize BPP, which looks supremely difficult (if possible at all). Finally, part 3 shows that even establishing the most relaxed version of the statement "MCSP is NP-complete" requires separating exponential time from randomized polynomial time, a separation that appears inaccessible with current proof techniques.

MCSP **is not hard under "local" reductions.** Many NP-complete problems are still complete under polynomial-time reductions with severe-looking restrictions, such as reductions which only need $O(\log^c n)$ time to compute an arbitrary bit of the output. Let $t : \mathbb{N} \to \mathbb{N}$; think of $t(n)$ as $n^{1-\varepsilon}$ for some $\varepsilon > 0$.

Definition 1.1. An algorithm $R : \Sigma^\star \times \Sigma^\star \to \{0,1,\star\}$ is a $\mathsf{TIME}(t(n))$ *reduction from L to L'* if there is a constant $c \geq 1$ such that for all $x \in \Sigma^\star$,

- $R(x,i)$ has random access to x and runs in $O(t(|x|))$ time for all $i \in \{0,1\}^{\lceil 2c \log_2 |x| \rceil}$,
- there is an $\ell_x \leq |x|^c + c$ such that $R(x,i) \in \{0,1\}$ for all $i \leq \ell_x$, and $R(x,i) = \star$ for all $i > \ell_x$, and
- $x \in L \iff R(x,1) \cdot R(x,2) \cdots R(x,\ell_x) \in L'$.

(Note that $\star$ denotes an "out of bounds" character to mark the end of the output.) That is, the overall reduction outputs strings of polynomial length, but any desired bit of the output can be printed in $O(t(n))$ time. $\mathsf{TIME}(n^{o(1)})$ reductions are powerful enough for almost all NP-completeness results, which have

[1] After learning of our preliminary results, Allender, Holden, and Kabanets [8] found an alternative proof of the consequence $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$.

"local" structure transforming small pieces of the input to small pieces of the output.[2] More precisely, an $O(n^k)$-time reduction R from L to L' is a *projection* if there is a polynomial-time algorithm A that, given $i = 1, \ldots, n^k$ in binary, A outputs either a fixed bit (0 or 1) which is the i-th bit of $R(x)$ for all x of length n, or a $j = 1, \ldots, n$ with $b \in \{0,1\}$ such that the i-th bit of $R(x)$ (for all x of length n) equals $b \cdot x_j + (1-b) \cdot (1-x_j)$. Skyum and Valiant [23] observed that almost all NP-complete problems are also complete under projections. So for example, we have:

Proposition 1.2 ([23, 22])**.** SAT, Vertex Cover, Independent Set, Hamiltonian Path*, and* 3-Coloring *are* NP*-complete under* $\mathsf{TIME}(\text{poly}(\log n))$ *reductions.*

In contrast to the above, we prove that MCSP is *not* complete under $\mathsf{TIME}(n^{1/3})$ reductions. Indeed there is no local reduction from even the simple language PARITY to MCSP.

Theorem 1.3. *For every $\delta < 1/2$, there is no* $\mathsf{TIME}(n^{\delta})$ *reduction from* PARITY *to* MCSP. *As a corollary,* MCSP *is not* $\mathsf{AC0}[2]$*-hard under* $\mathsf{TIME}(n^{\delta})$ *reductions.*[3]

This establishes that MCSP cannot be "locally" NP-hard in the way that many canonical NP-complete problems are known to be.

This theorem for local reductions can be applied to randomized local reductions as well, though with a slightly weaker result.

Definition 1.4. An algorithm $R : \Sigma^\star \times \Sigma^\star \times \Sigma^\star \to \{0,1,\star\}$ is a $\mathsf{TIME}(t(n))$ *randomized reduction from L to L'* if there are constants $c \geq 1, p < 1/2$ such that for all $x \in \Sigma^\star$,

- $R(x,r,i)$ has random access to x and runs in $O(t(|x|))$ time for all $i \in \{0,1\}^{\lceil 2c \log_2 |x| \rceil}$,
- the number of random bits $|r|$ used is at most $O(t(|x|))$,
- there is an $\ell_x \leq |x|^c + c$ such that $R(x,r,i) \in \{0,1\}$ for all $i \leq \ell_x$, and $R(x,r,i) = \star$ for all $i > \ell_x$,
- $x \in L \implies \Pr_r[R(x,r,1) \cdot R(x,r,2) \cdots R(x,r,\ell_x) \in L'] \geq 1-p$, and
- $x \notin L \implies \Pr_r[R(x,r,1) \cdot R(x,r,2) \cdots R(x,r,\ell_x) \in L'] \leq p$.

Under this definition, a randomized local reduction treats the random bits r as auxiliary input, and will map each x,r pair to its own instance such that the probability over all possible random strings r that R maps x,r correctly (that is, x,r is mapped to an instance of L' if and only if $x \in L$) is at least $1-p$. The case of a bitwise reduction, in which the reduction maps each x to a single MCSP instance but each bit i is printed correctly with probability $1-p$, can be reduced to the above definition by running the reduction $O(\log \ell)$ times and taking the majority bit. Since ℓ is polynomial in $|x|$ this only adds a logarithmic factor to the run time of the algorithm. Furthermore, for any value of r the probability that there is an i such that $R(x,r,i)$ maps to the wrong bit is constant by a union bound, which means that most random strings will print the correct bit for all values of i.

Under this definition of randomized local reductions, the theorem generalizes for slightly smaller polynomial time.

Theorem 1.5. *For every $\delta < 1/5$, there is no* $\mathsf{TIME}(n^{\delta})$ *randomized reduction from* PARITY *to* MCSP.

---

[2]We say "almost all NP-completeness results" because one potential counterexample is the typical reduction from Subset Sum to Partition: two numbers in the output of this reduction require taking the *sum of all numbers* in the input Subset Sum instance. Hence the straightforward reduction does not seem to be computable even in $2^{n^{o(1)}}$-size $\mathsf{AC0}$.

[3]Dhiraj Holden and Chris Umans (personal communication) proved independently that there is no $\mathsf{TIME}(\text{poly}(\log n))$ reduction from SAT to MCSP unless $\mathsf{NEXP} \subseteq \Sigma_2\mathsf{P}$.

Hardness under stronger reducibilities. For stronger reducibility notions than subpolynomial time, we do not yet have unconditional non-hardness results for MCSP. (Of course, a proof that MCSP is not NP-complete under polynomial-time reductions would immediately imply $\mathsf{P} \neq \mathsf{NP}$.) Nevertheless, we can still prove interesting complexity consequences assuming the NP-hardness of MCSP under these sorts of reductions.

Theorem 1.6. *If* MCSP *is* NP*-hard under polynomial-time reductions, then* $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P}/\mathrm{poly}$. *Consequently,* $\mathsf{EXP} \neq \mathsf{ZPP}$.

Theorem 1.7. *If* MCSP *is* NP*-hard under logspace reductions, then* $\mathsf{PSPACE} \neq \mathsf{ZPP}$.

Theorem 1.8. *If* MCSP *is* NP*-hard under logtime-uniform* $\mathsf{AC0}$ *reductions, then* $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$ *and* $\mathsf{E} \not\subset \text{i.o.-}\mathsf{SIZE}(2^{\delta n})$ *for some* $\delta > 0$. *As a consequence,* $\mathsf{P} = \mathsf{BPP}$ *also follows.*

That is, the *difficulty* of computing circuit complexity would imply *lower bounds*, even in the most general setting (there are *no* restrictions on the polynomial-time reductions here, in contrast with Kabanets and Cai [20]). We conjecture that the consequence of Theorem 1.6 can be strengthened to $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$, and that MCSP is (unconditionally) not NP-hard under uniform $\mathsf{AC0}$ reductions.

$\Sigma_2\mathsf{P}$-hardness for nondeterministic MCSP implies circuit lower bounds. Intuitively, the difficulty of solving MCSP via uniform algorithms should be related to circuit lower bounds against functions defined by uniform algorithms. That is, our intuition is that "MCSP is NP-complete" (under polynomial-time reductions) implies circuit lower bounds. We have not yet shown a result like this (but come close with $\mathsf{EXP} \neq \mathsf{ZPP}$ in Theorem 1.6). However, we can show that $\Sigma_2\mathsf{P}$-completeness for the *nondeterministic* version of MCSP would imply $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$.

In the Nondeterministic Minimum Circuit Size Problem (NMCSP), we are given $\langle T, k \rangle$ as in MCSP, but now we want to know if T denotes a Boolean function with *nondeterministic* circuit complexity at most k. It is easy to see that NMCSP is in $\Sigma_2\mathsf{P}$: existentially guess a circuit C with a "main" input and "auxiliary" input, existentially evaluate C on all 2^ℓ inputs x for which $T(x) = 1$, then universally verify on all 2^ℓ inputs y satisfying $T(y) = 0$ that no auxiliary input makes C output 1 on y.

We can show that if NMCSP is hard even for Merlin-Arthur games, then circuit lower bounds follow.

Theorem 1.9. *If* NMCSP *is* MA*-hard under polynomial-time reductions, then* $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$.

Vinodchandran [24] studied NMCSP for *strong* nondeterministic circuits, showing that a "natural" reduction from SAT or Graph Isomorphism to this problem would have several interesting implications.

1.1 Intuition

The MCSP problem is a special kind of "meta-algorithmic" problem, where the input describes a function (and a complexity upper bound) and the goal is to essentially compute the circuit complexity of the function. That is, like many of the central problems in theory, MCSP is a problem about computation itself.

In this paper, we apply many tools from the literature to prove our results, but the key idea is to exploit the meta-algorithmic nature of MCSP directly in the assumed reductions to MCSP. We take advantage of the fact that instances of MCSP are written in a rather non-succinct way: the entire truth table of the function is provided. (This observation was also used by Kabanets and Cai [20], but not to the same effect.)

For the simplest example of the approach, let L be a unary (tally) language, and suppose there is a $\mathsf{TIME}(\mathrm{poly}(\log n))$ reduction R from L to MCSP. The outputs of R are pairs $\langle T, k\rangle$, where T is a truth table and k is the size parameter. Because every bit of R is computable in polylog time, it follows that each truth table T output by R can in fact be described by a polylogarithmic-size circuit specifying the length of the input instance of L, and the mechanics of the polylog-time reduction used to compute a given bit of R. Therefore the circuit complexities of *all outputs of* R are at most polylogarithmic in n (the input length). Furthermore, the size parameters k in the outputs of R on n-bit inputs are at most $\mathrm{poly}(\log n)$, otherwise the MCSP instance is trivially a *yes* instance. That is, the efficient reduction R itself yields a strong upper bound on the witness sizes of the outputs of R.

This ability to bound k from above by a small value based on the existence of an efficient reduction to MCSP is quite powerful. It can also be carried out for more complex languages. For example, consider a polylog-time reduction from PARITY to MCSP, where we are mapping n-bit strings to instances of MCSP. Given any polylog-time reduction from PARITY to MCSP, we can construct another polylog-time reduction which on every n-bit string always outputs the *same* circuit size parameter k_n. That is, we can turn any polylog-time reduction into a *natural reduction* in the sense of Kabanets and Cai [20], and apply their work to *general* reductions. (The basic idea is to answer "no" to every bit query of the polylog-time reduction when computing k, and to then "pad" a given PARITY instance with a few strategically placed zeroes, so that it always satisfies those "no" answers. Since k is small and the reduction can make very few queries the amount of padding needed is relatively small.)

Several of our theorems have the form that, if computing circuit complexity is NP-hard (or nondeterministic circuit complexity is $\Sigma_2\mathsf{P}$-hard), then circuit lower bounds follow. This is intriguing to us, as one also expects that *efficient algorithms* for computing circuit complexity also lead to lower bounds! (For example, [20, 18, 26] show that polynomial-time algorithms for MCSP in various forms would imply circuit lower bounds against EXP and/or NEXP.) If a circuit lower bound can be proved to follow from assuming MCSP is NP-intermediate (or NMCSP is $\Sigma_2\mathsf{P}$-intermediate), perhaps we can prove circuit lower bounds unconditionally without necessarily resolving the complexity of MCSP.

2 Preliminaries

For simplicity, all languages are over $\{0,1\}$. We assume knowledge of the basics of complexity theory [9]. Here are a few (perhaps) non-standard notions we use. For a function $s : \mathbb{N} \to \mathbb{N}$, $\mathrm{poly}(s(n))$ is shorthand for $O(s(n)^c)$ for some constant c, and $\tilde{O}(s(n))$ is shorthand for $s(n) \cdot \mathrm{poly}(\log n)$. Define $\mathsf{SIZE}(s(n))$ to be the class of languages computable by a circuit family of size $O(s(n))$. In some of our results, we apply the well-known PARITY lower bound of Håstad:

Theorem 2.1 (Håstad [15])**.** *For every $k \geq 2$,* PARITY *cannot be computed by circuits with AND, OR, and NOT gates of depth k and size* $2^{o\left(n^{1/(k-1)}\right)}$.

Machine model. The machine model used in our results may be any model with random access to the input via addressing, such as a random-access Turing machine. The main component we want is that the "address" of the bit/symbol/word being read at any step is stored as a readable and writable binary integer.

A remark on subpolynomial reductions. In Definition 1.1 we defined subpolynomial-time reductions to output out-of-bounds characters which denote the end of an output string. We could also have defined our reductions to output a string of length $2^{\lceil c \log_2 n \rceil}$ on an input of length n, for some fixed constant $c \geq 1$. This makes it easy for the reduction to know the "end" of the output. We can still compute the length ℓ of the output in $O(\log \ell)$ time via Definition 1.1, by performing a doubling search on the indices i to find one $\star$ (trying the indices $1,2,4,8$, etc.), then performing a binary search for the first $\star$. The results in this paper hold for either reduction model (but the encoding of MCSP may have to vary in trivial ways, depending on the reduction notion used).

Encoding MCSP. Let $y_1, \ldots, y_{2^\ell} \in \{0,1\}^\ell$ be the list of k-bit strings in lexicographic order. Given $f : \{0,1\}^\ell \to \{0,1\}$, the truth table of f is defined to be $tt(f) := f(y_1)f(y_2)\cdots f(y_{2^\ell})$.

The truth table of a circuit is the truth table of the function it computes. Let $T \in \{0,1\}^\star$. The *function encoded by* T, denoted as f_T, is the function satisfying $tt(f_T) = T0^{2^k - |T|}$, where k is the minimum integer satisfying $2^k \geq T$. The *circuit complexity of* T, denoted as $CC(T)$, is simply the minimum number of gates of any circuit computing f_T.

There are several possible encodings of MCSP we could use. The main point we wish to stress is that it is possible to encode the circuit size parameter k in essentially unary or in binary, and our results remain the same. (This is important, because some of our proofs superficially seem to rely on a short encoding of k.) We illustrate our point with two encodings, both of which are suitable for the reduction model of Definition 1.1. First, we may define MCSP to be the set of strings Tx where $|T|$ is the largest power of two satisfying $|T| < |Tx|$ and $CC(f_T) \leq |x|$; we call this a *unary encoding* because k is effectively encoded in unary. (Note we cannot detect if a string has the form 1^k in logtime, so we shall let any k-bit string x denote the parameter k. Further note that, if the size parameter $k > |T|/2$, then the instance would be trivially a yes-instance. Hence this encoding captures the "interesting" instances of the problem.) Second, we may define MCSP to be the set of binary strings Tk such that $|T|$ is the largest power of two such that $|T| < |Tk|$, k is written in binary (with most significant bit 1) and $CC(f_T) \leq k$. Call this the *binary encoding*.

Proposition 2.2. *There are* $\mathsf{TIME}(\text{poly}(\log n))$ *reductions between the unary encoding of* MCSP *and the binary encoding of* MCSP.

The proof is a simple exercise, in Appendix A. More points on encoding MCSP for these reductions can be found there as well.

Another variant of MCSP has the size parameter fixed to a large value; this version has been studied extensively in the context of KT-complexity [3, 5]. Define MCSP$'$ to be the version with circuit size parameter set to $|T|^{1/2}$, that is, $\text{MCSP}' := \{T \mid CC(T) \leq |T|^{1/2}\}$. To the best of our knowledge, all theorems in this paper hold for MCSP$'$ as well; indeed most of the proofs only become simpler for this case.

A simple lemma on the circuit complexity of substrings. We also use the fact that for any string T, the circuit complexity of an arbitrary substring of T can be bounded via the circuit complexity of T.

Lemma 2.3 ([26]). *There is a universal $c \geq 1$ such that for any binary string T and any substring S of T, $CC(f_S) \leq CC(f_T) + c\log|T|$.*

Proof. Let c' be sufficiently large in the following. Let k be the minimum integer satisfying $2^k \geq |T|$, so the Boolean function f_T representing T has truth table $T0^{2^k-|T|}$. Suppose C is a size-s circuit for f_T. Let S be a substring of $T = t_1 \cdots t_{2^k} \in \{0,1\}^{2^k}$, and let $A, B \in \{1,\ldots,2^k\}$ be such that $S = t_A \cdots t_B$. Let $\ell \leq k$ be a minimum integer which satisfies $2^\ell \geq B - A$. We wish to construct a small circuit D with ℓ inputs and truth table $S0^{2^\ell-(B-A)}$. Let $x_1,\ldots,x_{2^\ell}$ be the ℓ-bit strings in lexicographic order. Our circuit D on input x_i first computes $i+A$; if $i+A \leq B-A$ then D outputs $C(x_{i+A})$, otherwise D outputs 0. Note there are circuits of $c' \cdot n$ size for addition of two n-bit numbers (this is folklore). Therefore in size at most $c' \cdot k$ we can, given input x_i of length ℓ, output $i+A$. Determining if $i+A \leq B-A$ can be done with $(c' \cdot \ell)$-size circuits. Therefore D can be implemented as a circuit of size at most $s + c'(k+\ell+1)$. To complete the proof, let $c \geq 3c'$. ☐

3 MCSP and sub-polynomial time reductions

In this section, we prove the following impossibility results for NP-hardness of MCSP.

Reminder of Theorem 1.3. *For every $\delta < 1/2$, there is no $\mathsf{TIME}(n^\delta)$ reduction from* PARITY *to* MCSP. *As a corollary,* MCSP *is not* $\mathsf{AC0}[2]$*-hard under* $\mathsf{TIME}(n^\delta)$ *reductions.*

Reminder of Theorem 1.5. *For every $\delta < 1/5$, there is no $\mathsf{TIME}(n^\delta)$ randomized reduction from* PARITY *to* MCSP.

3.1 Deterministic reductions

The proof of the deterministic case has the following outline. First we show that there are $\mathrm{poly}(\log n)$-time reductions from PARITY to itself which can "insert $\mathrm{poly}(n)$ zeroes" into a PARITY instance. Then, assuming there is a $\mathsf{TIME}(n^\delta)$ reduction from PARITY to MCSP, we use the aforementioned zero-inserting algorithm to turn the reduction into a "natural reduction" (in the sense of Kabanets and Cai [20]) from PARITY to MCSP, where the circuit size parameter k output by the reduction depends only on the input length n. Next, we show how to bound the value of k from above by $\tilde{O}(n^\delta)$, by exploiting naturalness. Then we use this bound on k to construct a depth-three circuit family of size $2^{\tilde{O}(n^\delta)}$, and appeal to Håstad's $\mathsf{AC0}$ lower bound for PARITY for a contradiction.

We start with a simple $\mathrm{poly}(\log n)$-time reduction for padding a string with zeroes in a $\mathrm{poly}(n)$-size set of prescribed bit positions. Let $S \in \mathbb{Z}^\ell$ for a positive integer ℓ. We say S is *sorted* if $S[i] < S[i+1]$ for all $i = 1,\ldots,\ell-1$.

Proposition 3.1. *Let $p(n)$ be a polynomial. There is an algorithm A such that, given x of length n, a sorted tuple $S = (i_1,\ldots,i_{p(n)})$ of indices from $\{1,\ldots,n+p(n)\}$, and a bit index $j = 1,\ldots,p(n)+n$, $A(x,S,j)$ outputs the j-th bit of the string x' obtained by inserting zeroes in the bit positions $i_1,i_2,\ldots,i_{p(n)}$ of x. Furthermore, $A(x,S,j)$ runs in $O(\log^2 n)$ time on x of length n.*

Proof. Given x of length n, a sorted $S = (i_1, \ldots, i_p) \in \{1, \ldots, n+p\}^p$, and an index $j = 1, \ldots, n+p$, first A checks if $j \in S$ in $O(\log^2 n)$ time by binary search, comparing pairs of $O(\log n)$-bit integers in $O(\log n)$ time. If yes, then A outputs 0. If no, there are two cases: either (1) $j < i_1$, or (2) $i_k < j$ for some $k = 1, \ldots, p$. In case (1), A simply outputs x_j. In case (2), A outputs x_{j-k}. (Note that computing $j - k$ is possible in $O(\log n)$ time.) It is easy to verify that the concatenation of all outputs of A over $j = 1, \ldots, |x| + p$ is the string x but with zeroes inserted in the bit positions $i_1, \ldots, i_p$. □

Let $t(n) = n^{1-\varepsilon}$ for some $\varepsilon > 0$. The next step is to show that a $\mathsf{TIME}(t(n))$ reduction from PARITY to MCSP can be turned into a *natural* reduction, in the following sense:

Definition 3.2 (Kabanets-Cai [20])**.** A reduction from a language L to MCSP is *natural* if the size of all output instances and the size parameters k depend only on the length of the input to the reduction.

The main restriction in the above definition is that the size parameter k output by the reduction does not vary over different inputs of length n.

Claim 3.3. *If there is a* $\mathsf{TIME}(t(n))$ *reduction from* PARITY *to* MCSP*, then there is a* $\mathsf{TIME}(t(n)\log^2 n)$ *natural reduction from* PARITY *to* MCSP*. Furthermore, the value of k in this natural reduction is* $\tilde{O}(t(n))$.

Proof. By assumption, we can choose n large enough to satisfy $t(2n)\log(2n) \ll n$. We define a new (natural) reduction R' from PARITY to MCSP as follows.

> $R'(x,i)$ begins by gathering a list of the bits of the input that affect the size parameter k of the output, for a hypothetical $2n$-bit input which has zeroes in the positions read by R. This works as follows. We simulate the $\mathsf{TIME}(t(n))$ reduction R from L to MCSP on the output indices corresponding to bits of the size parameter k, *as if* R is reading an input x' of length $2n$. When R attempts to read a bit of the input, record the index i_j requested in a list S, and continue the simulation as if the bit at position i_j is a 0. Since the MCSP instance is polynomial in size, k written in binary is at most $O(\log n)$ bits (otherwise we may simply output a trivial "yes" instance), so the number of indices of the output that describe k is at most $O(\log n)$ in the binary encoding. It follows that the size parameter k in the output depends on at most $t(2n)\log(2n)$ bits of the (hypothetical) $2n$-bit input. Therefore $|S| \le t(2n)\log(2n)$. Sort $S = (i_1, \ldots, i_{|S|})$ in $O(t(n)\log^2 n)$ time, and remove duplicate indices.
>
> R' then simulates the $\mathsf{TIME}(t(n))$ reduction $R(x,i)$ from PARITY to MCSP. However, whenever an input bit j of x is requested by R, if $j \le n + |S|$ then run the algorithm $A(x,S,j)$ from Proposition 3.1 to instead obtain the j-th bit of the $O(n+|S|)$-bit string x' which has zeroes in the bit positions in the sorted tuple S. Otherwise, if $j > n + |S|$ and $j \le 2n$ then output 0, and if $j > 2n$ then output $\star$ (out of bounds). Since the algorithm of Proposition 3.1 runs in $O(\log^2 n)$ time, this step of the reduction takes $O(t(n)\log^2 n)$ time.

That is, the reduction R' first looks for all the bits in a $2n$-bit input that affect the output size parameter k in the reduction R, assuming the bits read are all 0. Then R' runs R on a simulated $2n$-bit string x' for which all those bits are zero (and possibly more at the end, to enforce $|x'| = 2n$). Since the parity of x'

equals the parity of x, the MCSP instance output by R' is a yes-instance if and only if x has odd parity. However for the reduction R', the output parameter k is now a function of only the input length; that is, R' is natural.

Now let us argue for an upper bound on k. Define a function $f(i)$ which computes $z := 0^n$, then runs and outputs $R'(z,i)$. Since instances of MCSP are not exact powers of 2, assume that this function adds zero padding at the end, outputting zero when the index is out of range. Some prefix of the truth table of f, $tt(f)$, is therefore an instance of MCSP. Since R' is natural, the value of k appearing in $tt(f)$ is the *same* as the value of k for all length-n instances of PARITY.

However, the circuit complexity of f is *small*: on any i, $R'(0^n,i)$ can be computed in time $O(t(n)\log^2 n)$. Therefore the circuit complexity of f is at most some s which is $\tilde{O}(t(n))$. In particular, the $\mathsf{TIME}(t(n)\log^2 n)$ reduction can be efficiently converted to a circuit, with any bit of the input 0^n efficiently computed in $O(\log n)$ time at every request (the only thing to check is that the index requested does not exceed n). As the function f has $CC(f) \le s$, by Lemma 2.3 the truth table T corresponding to the instance of MCSP in $tt(f)$ has $CC(T) \le cs$ as well for some constant c.

Since 0^n has even parity, the truth table of f is not in MCSP. This implies that the value of k in the instance $tt(f)$ must be *less than* $cs = \tilde{O}(t(n))$. Therefore the value of k fixed in the reduction from PARITY to MCSP must be at most $\tilde{O}(t(n)\log^2 n)$. □

Finally, we can complete the proof of Theorem 1.3:

Proof of Theorem 1.3. Suppose that PARITY has a $\mathsf{TIME}(n^\delta)$ reduction from PARITY to MCSP, for some $\delta < 1/2$. Then by Claim 3.3, there is a reduction R' running in time $\tilde{O}(n^\delta \log^2 n)$ for which the witness circuit has size at most $\tilde{O}(n^\delta)$. Such an algorithm can be used to construct a depth-three OR-AND-OR circuit of size $2^{\tilde{O}(n^\delta)}$ that solves PARITY: the top OR at the output has incoming wires for all possible $2^{\tilde{O}(n^\delta)}$ existential guesses for the witness circuit C, the middle AND has incoming wires for all $n^{O(1)}$ bits i of the truth table produced by the reduction R', and the remaining deterministic computation on $n + \tilde{O}(n^\delta)$ bits, checking whether $C(i) = R'(x,i)$ on all inputs i, is computable with a CNF (AND of ORs) of size $2^{\tilde{O}(n^\delta)}$. Therefore, the assumed reduction implies that PARITY has depth-three $\mathsf{AC0}$ circuits of size $2^{\tilde{O}(n^\delta)}$. For $\delta < 1/2$, this is false by Håstad (Theorem 2.1). □

3.2 Randomized reductions

Now we turn to proving that PARITY does not have ultra-efficient *randomized* reductions to MCSP (Theorem 1.5). The proof of Theorem 1.5 mostly follows the same form as the deterministic case. Starting with a reduction from PARITY to MCSP, we add a poly($\log n$)-time reduction from PARITY to itself that can insert "poly(n)" zeroes to a PARITY instance, in such a way that the reduction is converted into a "natural reduction." (Recall that a natural reduction to MCSP creates instances whose size parameter k only depends on the length of the input string.) From this reduction, we can construct a faster algorithm for PARITY which can be converted into a constant depth circuit family that contradicts Håstad's $\mathsf{AC0}$ lower bound for PARITY. However, there are new complications that appear in the randomized case.

First, for every fixed random string r, a $\mathsf{TIME}(t(n))$ reduction R (with randomness r) can only depend on $O(t(n))$ bits of the input. However, since every random string can depend on a different set of input bits, a deterministic padding reduction *cannot* fix the value of k in every case. To solve this problem, it is enough to make the padding reduction randomized as well, and argue that k is fixed to a specific value with high probability. To this end, we begin with a proposition:

Proposition 3.4. *Let $p(n)$ be a polynomial. There is an algorithm A' which, given x of length n, r of length $\log_2(p(n))$ (interpreted as an integer), and a bit index $j = 1, \cdots, p(n) \cdot n$, $A'(x, r, j)$ outputs the j-th bit of the string x' obtained by inserting $p(n) - 1$ zeroes between each consecutive pair of bits of the string x, r zeroes before the first bit, and $p(n) - (r + 1)$ bits after the last bit. Furthermore, $A(x, r, j)$ runs in $O(\log(n) + \log(p(n)))$ time.*

Proof. Given x of length n, r of length $\log(p(n))$, and $j \le n \cdot p(n)$, compute j_1, j_2 such that $j = j_1 \cdot p(n) + j_2$. If $j_2 = r$, then output the j_1-th bit of x. Otherwise, output 0. Since division (with remainder) of unsigned integers can be computed in time linear in the number of bits, the algorithm runs in $O(\log(p(n)) + \log(n))$ time. □

Essentially, this reduction algorithm A' takes the original instance and embeds it uniformly spaced into a string of $n(p(n) - 1)$ zeroes, and the randomness r determines where in the all-zeroes string to start inserting bits of the original input. We will crucially use the following fact: *over all choices of r, the probability that a random output bit of R' comes from the original input x is $1/p(n)$*. This is true because in every substring of x' of $p(n)$ bits, there is exactly one bit of the original input x, and the position of that input bit is uniformly distributed across all $p(n)$ indices by the value of r.

Claim 3.5. *Let $c \in (0, 1)$. If there is a $\mathsf{TIME}(n^c)$ randomized reduction R from* PARITY *to* MCSP*, then there is a $\mathsf{TIME}(\tilde{O}(n^{c/(1-c)}))$ natural randomized reduction R' from* PARITY *to* MCSP*. Furthermore, the value of the size parameter k in this natural reduction is $\tilde{O}(n^{c/(1-c)})$.*

Again, this result follows from the deterministic case (Claim 3.3), though the instance requires much more padding. Also, since the padding is randomized and does not depend on the algorithm R, the reduction R' does not have to simulate R to determine which of the input bits the value of k depends on.

Proof. Suppose there is a $\mathsf{TIME}(n^c)$ randomized reduction R from PARITY to MCSP. Define a reduction $R'(x, r_1 r_2, i)$ which simply simulates and returns the output of $R(x', r_1, i)$, where

$$x' = A'(x, r_2, 1) \cdot A'(x, r_2, 2) \cdots A'(x, r_2, |x| \cdot p(|x|))$$

is the PARITY instance produced by running the randomized padding algorithm A' from Proposition 3.4 on randomness r_2, and $p(n)$ is a polynomial to be specified later. Since R is a local reduction, the string x' is implicit, and does not need to be stored in memory: whenever R attempts to read the j-th bit of x' (for some j), R' computes $A'(x, r_2, j)$ and continues the simulation using this value as the j-th bit of x'. Since A' runs in logarithmic time, the reduction R' runs in time $O(|x|^c \cdot p(|x|)^c \log^d(|x|))$ for some constant $d > 0$.

Let $n = |x|$, and fix r_1, the randomness for reduction R. The probability that a given bit read by R is not a padding bit is $1/p(n)$; since r_1 and r_2 are independent, this is true for every bit read by R. For a fixed string r_1, the algorithm R becomes a standard deterministic reduction; in that case, the value of

k depends on at most $O((n \cdot p(n))^c \log(n \cdot p(n)))$ input bits. By a union bound, the probability over all values of r_2 that at least one read bit is not a padding bit is at most

$$\frac{O((n \cdot p(n))^c \log(n \cdot p(n)))}{p(n)} \leq O(n^c \cdot p(n)^{c-1} \cdot \log(n)).$$

Setting $p(n) = n^{\frac{1}{1-c}-1} \cdot \log^e(n)$ for a constant $e > 1$ (using $(1/(1-c)-1)\log n + e \log\log n)$ random bits in the PARITY reduction), the probability that the value of k depends on a non-padding bit is at most $O(1)/\log^{e-1-ce} n$. Noting that $c < 1$, and setting e to be sufficiently large, the reduction R' has a high probability of fixing the value of k to a constant, and the probability that the reduction both fixes k to a constant and maps to a correct instance of MCSP is at least $1 - p - o(1) > 1 - p'$ for some constant $p' < 1/2$. Therefore this randomized reduction is natural.

Under the above setting of $p(n)$, the instance x' produced by the padding reduction A' has size $n \cdot p(n) = n^{1/(1-c)} \log^e(n)$, and running the given reduction R on x' takes time $\tilde{O}(|x'|^c) = \tilde{O}(n^{c/(1-c)})$.

Given the natural reduction R', define a family of functions $f(i) = R'(0^n, r_n, i)$, where r_n is a random string that maps 0^n to a *no* instance of MCSP. Since at least $1 - p'$ of all random strings meet this requirement, such an r_n must exist. As in the deterministic case, the circuit complexity of f is small, since $R'(0^n, r_n, i)$ can be computed in time $O(n^{c/(1-c)})$. Treating the randomness as hardwired advice, this reduction can be efficiently converted to a circuit of size $\tilde{O}(n^{c/(1-c)})$. As a result, the truth table T produced by the reduction R' on input $(0^n, r_n)$ has $CC(T) \leq \tilde{O}(n^{c/(1-c)})$; since the output of R' is not an instance of MCSP, we can conclude that the fixed value of k is at most $\tilde{O}(n^{c/(1-c)})$. □

Finally, we complete the proof of Theorem 1.5:

Proof of Theorem 1.5. Suppose that PARITY does have a $\mathsf{TIME}(n^\delta)$ randomized reduction to MCSP, for some $\delta < 1/5$. Then by applying Claim 3.5, there is a natural randomized reduction R' running in $\tilde{O}(n^{\delta/(1-\delta)})$ time for which k is at most $\tilde{O}\left(n^{\delta/(1-\delta)}\right)$. Let $\delta' = \delta/(1-\delta)$. If $\delta < 1/5$, then $\delta' < 1/4$, which means that there is a $\mathsf{TIME}(\tilde{O}(n^{\delta'}))$ randomized reduction.

We can construct a depth-five circuit for PARITY of size $2^{\tilde{O}(n^{\delta'})}$, as follows. For each possible $\tilde{O}(n^{\delta'})$-bit random string r used by the reduction, we can simulate the deterministic part of the $\mathsf{TIME}(\tilde{O}(n^{\delta'}))$ reduction by taking an OR over all possible witness circuits of size $\tilde{O}\left(n^{\delta/(1-\delta)}\right)$ which may compute the output of the reduction R' on input x, then taking an AND over all possible inputs to the circuit to verify the guess (the remaining deterministic computation on $\tilde{O}(n^{\delta'})$ bits can be computed with a CNF of size $2^{\tilde{O}(n^{\delta'})}$). This would be a depth-three circuit C_r of size $2^{\tilde{O}(n^{\delta/(1-\delta)})}$ solving the generated MCSP instance, for a fixed random string r.

There are $2^{\tilde{O}(n^{\delta'})}$ possible random strings to be given as input to the reduction. Using depth-three (OR-AND-OR) circuits for Approximate-Majority (Ajtai [2]) of size $2^{\tilde{O}(n^{\delta'})}$, we can compute the correct output over all possible depth-three circuits C_r. Merging the bottom OR of the Approximate Majority circuit with the top ORs of the C_r, we obtain a depth-five circuit computing PARITY. Therefore, the reduction implies that PARITY has depth-5 circuits of size $2^{\tilde{O}(n^{\delta'})}$. For $\delta' < 1/4$ this is false, by Håstad (Theorem 2.1). □

Remark 3.6. We used only the following properties of PARITY in the above proof: (a) one can insert zeroes into a string efficiently without affecting its membership in PARITY, (b) PARITY has trivial no-instances (strings of all zeroes), and (c) PARITY lacks small constant-depth circuits. We imagine that some of the ideas in the above proof may be useful for other "non-hardness" results in the future.

4 NP-hardness of MCSP implies lower bounds

We now turn to stronger reducibility notions, showing that even NP-hardness of MCSP under these reductions implies separation results that currently appear out of reach.

4.1 Consequences of NP-hardness under polytime and logspace reductions

Our two main results here are:

Reminder of Theorem 1.6. *If* MCSP *is* NP*-hard under polynomial-time reductions, then* $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P}/\text{poly}$. *Consequently,* $\mathsf{EXP} \neq \mathsf{ZPP}$.

Reminder of Theorem 1.7. *If* MCSP *is* NP*-hard under logarithmic-space reductions, then* $\mathsf{PSPACE} \neq \mathsf{ZPP}$.

These theorems follow from establishing that the NP-hardness of MCSP and small circuits for EXP implies $\mathsf{NEXP} = \mathsf{EXP}$. In fact, it suffices that MCSP is hard for only sparse languages in NP. (Recall that a language L is *sparse* if there is a c such that for all n, $|L \cap \{0,1\}^n| \leq n^c + c$.)

Theorem 4.1. *If every sparse language in* NP *has a polynomial-time reduction to* MCSP*, then* $\mathsf{EXP} \subset \mathsf{P}/\text{poly} \implies \mathsf{EXP} = \mathsf{NEXP}$.

Proof. Suppose that MCSP is hard for sparse NP languages under polynomial-time reductions, and that $\mathsf{EXP} \subset \mathsf{P}/\text{poly}$. Let $L \in \mathsf{NTIME}(2^{n^c})$ for some $c \geq 1$. It is enough to show that $L \in \mathsf{EXP}$.

Define the padded language $L' := \{x01^{2^{|x|^c}} \mid x \in L\}$. The language L' is then a sparse language in NP. By assumption, there is a polynomial-time reduction from L' to MCSP. Composing the obvious reduction from L to L' with the reduction from L' to MCSP, we have a $2^{c' \cdot n^c}$-time reduction R from n-bit instances of L to $2^{c' \cdot n^c}$-bit instances of MCSP, for some constant c'. Define the language

$$\text{BITS}_R := \{(x,i) \mid \text{the } i\text{-th bit of } R(x) \text{ is } 1\}.$$

BITS_R is clearly in EXP. Since $\mathsf{EXP} \subset \mathsf{P}/\text{poly}$, for some $d \geq 1$ there is a circuit family $\{C_n\}$ of size at most $n^d + d$ computing BITS_R on n-bit inputs.

Now, on a given instance x of L, the circuit $D(i) := C_{2|x|+c' \cdot |x|^c}(x,i)$ has $c' \cdot |x|^c$ inputs (ranging over all possible $i = 1, \ldots, 2^{c' \cdot |x|^c}$) and size at most $s(|x|) := (2+c')^d |x|^{cd} + d$, such that the output of $R(x)$ is $tt(D)$ (the truth table of D). Therefore, for every x, the truth tables output by $R(x)$ all have circuit complexity at most $e \cdot s(|x|)$ for some constant e, by Lemma 2.3. This observation leads to the following exponential-time algorithm for L.

> On input x, run the reduction $R(x)$, obtaining an exponential-size instance $\langle T,k\rangle$ of MCSP. If $k > e \cdot s(|x|)$ then *accept*. Otherwise, cycle through every circuit E of size at most k; if $tt(E) = T$ then *accept*. If no such E is found, *reject*.

Producing the truth table T takes exponential time, and checking all $2^{O(s(n)\log s(n))}$ circuits of size $O(s(n))$ on all polynomial-size inputs to the truth table also takes exponential time. As a result $L \in \mathsf{EXP}$, which completes the proof. □

The same argument can be used to prove collapses for other reducibilities. For example, swapping time for space in the proof of Theorem 4.1, we obtain:

Corollary 4.2. *If* MCSP *is* NP*-hard under logspace reductions, then* $\mathsf{PSPACE} \subset \mathsf{P/poly} \implies \mathsf{NEXP} = \mathsf{PSPACE}$.

Theorem 4.1 shows that complexity class separations follow from establishing that MCSP is NP-hard in the most general sense. We now prove Theorem 1.6, that NP-hardness of MCSP implies $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P/poly}$.

Proof of Theorem 1.6. By contradiction. Suppose MCSP is NP-hard and $\mathsf{EXP} = \mathsf{NP} \cap \mathsf{P/poly}$. Then $\mathsf{EXP} \subset \mathsf{P/poly}$ implies $\mathsf{NEXP} = \mathsf{EXP}$ by Theorem 4.1, but $\mathsf{NEXP} = \mathsf{EXP} \subseteq \mathsf{NP}$, contradicting the nondeterministic time hierarchy [27]. □

Theorem 1.7 immediately follows from the same argument as Theorem 1.6, applying Corollary 4.2.

We would like to strengthen Theorem 1.6 to show that the NP-hardness of MCSP actually implies *circuit* lower bounds such as $\mathsf{EXP} \not\subset \mathsf{P/poly}$. This seems like a more natural consequence: an NP-hardness reduction would presumably be able to print truth tables of high circuit complexity from no-instances of low complexity. (Indeed this is the intuition behind Kabanets and Cai's results concerning "natural" reductions [20].)

4.2 Consequences of NP-hardness under $\mathsf{AC0}$ reductions

Now we turn to showing consequences of the assumption that MCSP is NP-hard under uniform $\mathsf{AC0}$ reductions.

Reminder of Theorem 1.8. *If* MCSP *is* NP*-hard under logtime-uniform* $\mathsf{AC0}$ *reductions, then* $\mathsf{NP} \not\subset \mathsf{P/poly}$ *and* $\mathsf{E} \not\subset$ i.o.-$\mathsf{SIZE}(2^{\delta n})$ *for some* $\delta > 0$*. As a consequence,* $\mathsf{P} = \mathsf{BPP}$ *also follows.*

We will handle the two consequences in two separate theorems.

Theorem 4.3. *If* MCSP *is* NP*-hard under logtime* $\mathsf{AC0}$ *reductions, then* $\mathsf{NP} \subset \mathsf{P/poly} \implies \mathsf{NEXP} \subset \mathsf{P/poly}$.

Proof. The proof is similar in spirit to that of Theorem 4.1. Suppose that MCSP is NP-hard under logtime-uniform $\mathsf{AC0}$ reductions, and that $\mathsf{NP} \subset \mathsf{P/poly}$. Then $\Sigma_k\mathsf{P} \subset \mathsf{P/poly}$ for every $k \geq 1$.

Let $L \in \mathsf{NEXP}$; in particular, let $L \in \mathsf{NTIME}(2^{n^c})$ for some c. As in Theorem 4.1, define the sparse NP language $L' = \{x01^t \mid x \in L, t = 2^{|x|^c}\}$. By assumption, there is a logtime-uniform $\mathsf{AC0}$ reduction R from the sparse language L' to MCSP. This reduction can be naturally viewed as a $\Sigma_k\mathsf{P}$ reduction $S(\cdot,\cdot)$

from L to exponential-size instances of MCSP, for some constant k. In particular, $S(x,i)$ outputs the i-th bit of the reduction R on input $x01^t$, and S can be implemented in $\Sigma_k\mathsf{P}$, and hence in P/poly as well.

That is, for all inputs x, the string $S(x,1)\cdots S(x,2^{O(|x|^c)})$ is the truth table of a function with $\text{poly}(|x|)$-size circuits. Therefore by Lemma 2.3, the truth table of the MCSP instance being output on x must have a $\text{poly}(|x|)$-size circuit. We can then decide L in $\Sigma_{k+2}\mathsf{P}$ time: on an input x, existentially guess a circuit C of $\text{poly}(|x|)$ size, then for all inputs y to C, verify that $S(x,y) = C(y)$. The latter equality can be checked in $\Sigma_k\mathsf{P}$. As a result, we have $\mathsf{NEXP} \subseteq \Sigma_{k+2}\mathsf{P} \subset \mathsf{P}/\text{poly}$. □

Theorem 4.4. *If* MCSP *is* NP*-hard under* P*-uniform* AC0 *reductions, then there is a* $\delta > 0$ *such that* $\mathsf{E} \not\subset \text{i.o.-}\mathsf{SIZE}(2^{\delta n})$. *As a consequence,* $\mathsf{P} = \mathsf{BPP}$ *also follows from the same assumption (Impagliazzo and Wigderson [19]).*

Proof. Assume the opposite: that MCSP is NP-hard under P-uniform AC0 reductions and for every $\varepsilon > 0$, $\mathsf{E} \subset \text{i.o.-}\mathsf{SIZE}(2^{\varepsilon n})$. By Agrawal et al. [1] (Theorem 4.1), all languages hard for NP under P-uniform AC0 reductions are also hard for NP under P-uniform NC0 reductions. Therefore MCSP is NP-hard under P-uniform NC0 reductions. Since in an NC0 circuit all outputs depend on a constant number of input bits, the circuit size parameter k in the output of the reduction depends on only $O(\log n)$ input bits. Similar to the argument given for Claim 3.3, the NC0 reduction from PARITY to MCSP can be converted into a natural reduction. Therefore we may assume that the size parameter k in the output of the reduction is a function of only the length of the input to the reduction. (Nevertheless, we are not yet done at this point: the P-uniformity prevents us from upper-bounding the circuit complexity of the MCSP instances output by this reduction. Our additional hypothesis will let us do precisely that.)

Let R be a polynomial-time algorithm that on input 1^n produces a P-uniform NC0 circuit C_n on n inputs that reduces PARITY to MCSP. Fix c such that R runs in at most $n^c + c$ time and every truth table produced by the reduction is of length at most $n^c + c$. Define an algorithm R' as follows:

> On input (n,i,b), where n is a binary integer, $i \in \{1,\ldots,n^c+c\}$, and $b \in \{0,1\}$, run $R(1^n)$ to produce the circuit C_n, then evaluate $C_n(0^n)$ to produce a truth table T_n. If $b = 0$, output the i^{th} bit of C_n. If $b = 1$, output the i^{th} bit of T_n.

For an input (n,i,b), R' runs in time $O(n^c)$; when $m = |(n,i,b)|$, this running time is $2^{O(m)} \le n^{O(1)}$. By assumption, for every $\varepsilon > 0$, R' has circuits $\{D_m\}$ of size $O(2^{\varepsilon m}) \le O(n^{(c+2)2\varepsilon})$ for infinitely many input lengths m. This has two important consequences:

1. For every $\varepsilon > 0$ there are infinitely many input lengths $m = O(\log n)$ such that the size parameter k in the natural reduction from PARITY to MCSP is at most $n^{2\varepsilon}$ (or, the instance is trivial). To see this, first observe that 0^n is always a no-instance of PARITY, so $R(0^n)$ always maps to a truth table T_m of circuit complexity greater than $k(n)$ (for some $k(n)$). Since $R'(n,i,1)$ prints the i^{th} bit of $R(0^n)$, and the function $R'(n,\cdot,1)$ is computable with an $O(n^{(c+2)2\varepsilon})$-size circuit D_n, the circuit complexity of T_m is at most $O(n^{(c+2)2\varepsilon})$, by Lemma 2.3. Therefore the output size parameter k of $R(0^n)$ for these input lengths m is at most $O(n^{2\varepsilon})$.
2. On the *same* input lengths m for which k is $O(n^{(c+2)2\varepsilon})$, the *same* circuit D_m of size $O(n^{(c+2)2\varepsilon})$ can compute any bit of the NC0 circuit C_n that reduces PARITY to MCSP. This follows from simply setting $b = 0$ in the input of D_m.

The key point is that both conditions are simultaneously satisfied for infinitely many input lengths m, because both computations are made by the same $2^{O(n)}$-time algorithm R'. We use these facts to construct depth 3 circuits of size $2^{\tilde{O}(n^\gamma)}$ for $\gamma = 2\varepsilon(c+2)$ that computes parity for infinitely many input lengths m. As with previous proofs, the top OR has incoming wires for all possible circuits of size $\tilde{O}(n^\gamma)$ and the middle AND has wires for all possible bits i of the truth table produced by the reduction. At the bottom, construct a CNF of size $2^{\tilde{O}(n^\gamma)}$ computing the following $\mathsf{TIME}(\tilde{O}(n^\gamma))$ algorithm A.

> On input (x, C', i, D), where $n = |x|$ and D is an $O(n^\gamma)$ size circuit with $m = O(\log n)$ inputs:
>
> Assume D computes $R'(n,i,b)$ on inputs such that $m = |(n,i,b)|$. Evaluate D on n, $O(\log n)$ different choices of i, and $b = 0$, to construct the portion of the $\mathsf{NC0}$ circuit C_n that computes the size parameter k in the output of the reduction from PARITY to MCSP. Then, use this $O(\log n)$-size subcircuit to compute the value of k for the input length n.
>
> Check that C' is a circuit of size at most k on inputs of length $|i|$. We wish to verify that $C'(i)$ outputs the i^{th} bit of the truth table T_x produced by the $\mathsf{NC0}$ reduction on input x. We can verify this by noting that the i^{th} bit of T_x can also be computed in $O(n^\gamma)$ time, via D. Namely, produce the $O(1)$-size subcircuit that computes the i^{th} output bit of the $\mathsf{NC0}$ circuit C_n (by making a constant number of queries to D with $b = 0$). Then, simulate the resulting $O(1)$-size subcircuit on the relevant input bits of x to compute the i^{th} bit of T_x, and check that $C'(i)$ equals this bit. If all checks pass, *accept*, else *reject*.

Assuming the circuit D actually computes R', $A(x,C',i,D)$ checks whether $C'(i)$ computes the i-th bit of the $\mathsf{NC0}$ reduction. Taken as a whole, the depth-3 circuit computes PARITY correctly. For every $\varepsilon > 0$, there are infinitely many m such that the circuit size parameter k is at most $O(2^{\varepsilon m}) \le O(n^{(c+2)2\varepsilon})$, and the circuit D of size $O(2^{\varepsilon m}) \le O(n^{(c+2)2\varepsilon})$ exists. Under these conditions, the above algorithm A runs in $\tilde{O}(n^\gamma)$ time. As a result, for every $\gamma > 0$ we can find for infinitely many n such that has a corresponding depth-3 $\mathsf{AC0}$ circuit of $2^{\tilde{O}(n^\gamma)}$ size. Suppose we hardwire the $O(n^{2\varepsilon})$-size circuit D that computes R' into the corresponding $\mathsf{AC0}$ circuit, on input lengths n for which D exists. Then for all $\gamma > 0$, PARITY can be solved infinitely often with depth-3 $\mathsf{AC0}$ circuits of $2^{\tilde{O}(n^\gamma)}$ size, contradicting Håstad (Theorem 2.1). ☐

Proof of Theorem 1.8. Suppose MCSP is NP-hard under logtime-uniform $\mathsf{AC0}$ reductions. The consequence $\mathsf{E} \not\subset \mathsf{SIZE}(2^{\delta n})$ was already established in Theorem 4.4. The consequence $\mathsf{NP} \not\subset \mathsf{P/poly}$ follows immediately from combining Theorem 4.3 and Theorem 4.4. ☐

4.3 The hardness of nondeterministic circuit complexity

Finally, we consider the generalization of MCSP to the nondeterministic circuit model. Recall that a *nondeterministic circuit* C of size s takes two inputs, a string x on n bits and a string y on at most s bits. We say C computes a function $f : \{0,1\}^n \to \{0,1\}$ if for all $x \in \{0,1\}^n$,

$$f(x) = 1 \qquad \iff \qquad \text{there is a } y \text{ of length at most } s \text{ such that } C(x,y) = 1.$$

Observe that the Cook-Levin theorem implies that every nondeterministic circuit C of size s has an equivalent nondeterministic depth-two circuit C' of size $s \cdot \mathrm{poly}(\log s)$ (and *unbounded* fan-in). Therefore, it is no real loss of generality to define *Nondeterministic* MCSP (NMCSP) to be the set of all pairs $\langle T,k\rangle$ where T is the truth table of a function computed by a depth-two nondeterministic circuit of size at most k. This problem is in $\Sigma_2\mathsf{P}$: given $\langle T,k\rangle$, existentially guess a nondeterministic circuit C of size at most k, then for every x such that $T(x)=1$, existentially guess a y such that $C(x,y)=1$; for every x such that $T(x)=0$, universally verify that for all y, $C(x,y)=0$. However, NMCSP is not known to be $\Sigma_2\mathsf{P}$-hard. (The proof of Theorem 1.9 below will work for the depth-two version with unbounded fan-in, and the unrestricted version.)

Recall that it is *known* that MCSP for depth-two circuits is NP-hard [11, 7]. That is, the "deterministic counterpart" of NMCSP is known to be NP-hard.

Reminder of Theorem 1.9. *If* NMCSP *is* MA*-hard under polynomial-time reductions, then* $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$.

Proof. Suppose that NMCSP is MA-hard under polynomial-time reductions, and suppose that $\mathsf{EXP} \subset \mathsf{P}/\mathrm{poly}$. We wish to establish a contradiction. The proof is similar in structure to other theorems of this section (such as Theorem 4.1). Let $L \in \mathsf{MATIME}(2^{n^c})$, and define

$$L' = \{x01^{2^{|x|^c}} \mid x \in L\} \in \mathsf{MA}\,.$$

By assumption, there is a reduction R from L' to NMCSP that runs in polynomial time. Therefore for some constant d we have a reduction R' from L that runs in 2^{dn^c} time, and outputs 2^{dn^c}-size instances of NMCSP with a size parameter $s(x)$ on input x. Since R' runs in exponential time, and we assume EXP is in P/poly, there is a k such that for all x, there is a nondeterministic circuit $C((x,i),y)$ of $\le n^k+k$ size that computes the i-th bit of $R'(x)$. Therefore we know that $s(x) \le |x|^k + k$ on such instances (otherwise we can trivially *accept*). We claim that $L \in \mathsf{EXP}$, by the following algorithm:

Given x, run $R'(x)$ to compute $s(x)$. If $s(x) > |x|^k + k$ then *accept*.
    For all circuits C in increasing order of size up to $s(x)$,
        Initialize a table T of 2^{dn^c} bits to be all-zero.
        For all $i = 1,\ldots,2^{dn^c}$ and all $2^{s(x)}$ possible nondeterministic strings y,
            Check for each i if there is a y such that $C((x,i),y)=1$; if so, set $T[i]=1$.
        If $T = R'(x)$ then *accept*.
Reject (no nondeterministic circuit of size at most $s(x)$ was found).

Because $s(x) \le |x|^k + k$, the above algorithm runs in $2^{n^k \cdot \mathrm{poly}(\log n)}$ time and decides L. Therefore $L \in \mathsf{EXP}$. But this implies that $\mathsf{MAEXP} = \mathsf{EXP} \subset \mathsf{P}/\mathrm{poly}$, which contradicts the circuit lower bound of Buhrman, Fortnow, and Thierauf [10]. $\square$

5 Conclusion

We have demonstrated several formal reasons why it has been difficult to prove that MCSP is NP-hard. In some cases, proving NP-hardness would imply longstanding complexity class separations; in other cases, it is simply impossible to prove NP-hardness.

There are many open questions left to explore. Based on our study, we conjecture that:

- If MCSP is NP-hard under polynomial-time reductions then $\mathsf{EXP} \not\subset \mathsf{P}/\mathrm{poly}$. We showed that if MCSP is hard for sparse NP languages then $\mathsf{EXP} \neq \mathsf{ZPP}$; surely a reduction from SAT to MCSP would provide a stronger consequence. After the conference version of this paper appeared, Hitchcock and Pavan [17] showed that $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{SIZE}[2^{n^{\varepsilon}}]$ (for some $\varepsilon > 0$) is a consequence of the NP-hardness of MCSP under *truth-table reductions*, and Hirahara and Watanabe [16] have also shown that $\mathsf{EXP} \neq \mathsf{ZPP}$ follows from weaker hypotheses about the NP-completeness of MCSP.
- MCSP is (unconditionally) not NP-hard under logtime-uniform $\mathsf{AC0}$ reductions. Theorem 1.3 already implies that MCSP is not NP-hard under polylogtime-uniform $\mathsf{NC0}$ reductions. Perhaps this next step is not far away, since we already know that hardness under P-uniform $\mathsf{AC0}$ reductions implies hardness under P-uniform $\mathsf{NC0}$ reductions (by Agrawal et al. [1]).

It seems that we can prove that finding the minimum DNF for a given truth table is NP-hard, because of $2^{\Omega(n)}$ size lower bounds against DNFs [7]. Since there are $2^{\Omega(n^{\delta})}$ size lower bounds against $\mathsf{AC0}$, can it be proved that finding the minimum $\mathsf{AC0}$ circuit for a given truth table is $\mathsf{QuasiNP}$-hard? In general, can circuit lower bounds imply hardness results for circuit minimization?

While we have proven consequences if MCSP is NP-hard under general polynomial-time reductions, we have none if it is hard under polynomial-time randomized reductions. Is it possible to get a lower bound for this case as we were able to in the case of local reductions?

A Appendix: Unary and binary encodings of MCSP

We will describe our reductions in the reduction model with "out of bounds" errors (Definition 1.1). In that model, we may define *a unary encoding of* MCSP (T,k) to be Tx where $|T|$ is the largest power of two such that $|T| \leq |Tx|$, $k = |x|$. This encoding is sensible because we may assume without loss of generality that $k \leq 2|T|/\log|T|$: the size of a minimum circuit for a Boolean function on n inputs is always less than $2^{n+1}/n$ (for a reference, see for example [25, p.28–31]). Similarly, we defined MCSP (T,k) *in the binary encoding* to simply be Tk where $|T|$ is the largest power of two such that $|T| \leq |Tk|$.

Note that these encodings may be undesirable if one really wants to allow trivial yes instances in a reduction to MCSP where the size parameter k is too large for the instance to be interesting, or if one wants to allow T to have length other than a power of two. For those cases, the following binary encoding works: we can encode the instance (T,k) as the strings $T00k'$ such that k' is k written "in binary" over the alphabet $\{01,11\}$. There are also $\mathsf{TIME}(\mathrm{poly}(\log n))$ reductions to and from this encoding to the others above, mainly because k' has length $O(\log n)$.

Proposition A.1. *There are* $\mathsf{TIME}(\mathrm{poly}(\log n))$ *reductions between the unary encoding of* MCSP *and the binary encoding of* MCSP.

Proof. We can reduce from the binary encoding to the unary encoding as follows. Given an input y, perform a doubling search (probing positions 1, 2, 4, ..., 2^{ℓ}, etc.) until a $\star$ character is returned. Letting $2^{\ell} < |y|$ be the position of the last bit read, this takes $O(\log|y|)$ probes to the input. Then we may "parse" the input y into T as the first 2^{ℓ} bits, and integer k' as the remainder. To process the integer k', we begin by assuming $k' = 1$, then we read in $\log|y|)$ bits past the position 2^{ℓ}, doubling k' for each bit read and adding 1 when the bit read is 1, until $k' > |y|$ (in which case we do not have to read further: the instance is trivially yes) or we read a $\star$ (in which case we have determined the integer k'). Finally, if the bit position i requested is at most 2^ℓ, then we output the identical bit from the input Tk. If not, we print 1 if $i < 2^\ell + k + 1$, and $\star$ otherwise. The overall output of this reduction is $T1^{k'}$ where $k < |T|$. Since addition of $O(\log n)$ numbers can be done in $O(\log n)$ time, the above takes $\mathrm{poly}(\log n)$ time.

To reduce from the unary encoding to the binary encoding, we perform a doubling search on the input y as in the previous reduction, to find the largest ℓ such that $2^\ell < |y|$. Then we let the first 2^ℓ bits be T, and set the parameter $k = |y| - 2^\ell - 1$. (Finding $|y|$ can be done via binary search in $O(\log |y|)$ probes to the input.) From here, outputting the i-th bit of either T or k in the binary encoding is easy, since $|k| = O(\log |y|)$.

□

Acknowledgments. We thank Greg Bodwin and Brynmor Chapman for their thoughtful discussions on these results. We also thank Eric Allender, Oded Goldreich, and anonymous referees for helpful comments. We are also grateful to Eric for providing a preprint of his work with Dhiraj Holden.

References

[1] MANINDRA AGRAWAL, ERIC ALLENDER, RUSSELL IMPAGLIAZZO, TONIANN PITASSI, AND STEVEN RUDICH: Reducing the complexity of reductions. *Comput. Complexity*, 10(2):117–138, 2001. Preliminary version in STOC'97. [doi:10.1007/s00037-001-8191-1] 15, 18

[2] MIKLÓS AJTAI: Σ^1_1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. [doi:10.1016/0168-0072(83)90038-6] 12

[3] ERIC ALLENDER: When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *31th IARCS Conf. Found. of Software Tech. and Theoret. Comput. Sci. (FSTTCS'01)*, volume 2245 of *LNCS*, pp. 1–15. Springer, 2001. [doi:10.1007/3-540-45294-X_1] 7

[4] ERIC ALLENDER: Personal communication, 2014. 2

[5] ERIC ALLENDER, HARRY BUHRMAN, MICHAL KOUCKÝ, DIETER VAN MELKEBEEK, AND DETLEF RONNEBURGER: Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. Preliminary version in FOCS'02. [doi:10.1137/050628994] 2, 3, 7

[6] ERIC ALLENDER AND BIRESWAR DAS: Zero knowledge and circuit minimization. *Inform. and Comput.*, 2017. Preliminary versions in MFCS'14 and ECCC. [doi:10.1016/j.ic.2017.04.004] 2

[7] ERIC ALLENDER, LISA HELLERSTEIN, PAUL MCCABE, TONIANN PITASSI, AND MICHAEL SAKS: Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008. Preliminary version in CCC'06. [doi:10.1137/060664537] 2, 17, 18

[8] ERIC ALLENDER, DHIRAJ HOLDEN, AND VALENTINE KABANETS: The minimum oracle circuit size problem. *Comput. Complexity*, 26(2):469–496, 2017. Preliminary version in STACS'15. [doi:10.1007/s00037-016-0124-0] 3

[9] SANJEEV ARORA AND BOAZ BARAK: *Computational Complexity: A Modern Approach*. Cambridge Univ. Press, 2009. 6

[10] HARRY BUHRMAN, LANCE FORTNOW, AND THOMAS THIERAUF: Nonrelativizing separations. In *Proc. 13th IEEE Conf. on Computational Complexity (CCC'98)*, pp. 8–12. IEEE Comp. Soc. Press, 1998. [doi:10.1109/CCC.1998.694585] 17

[11] SEBASTIAN CZORT: The complexity of minimizing disjunctive normal form formulas, 1999. Master's Thesis, University of Aarhus. 2, 17

[12] VITALY FELDMAN: Hardness of approximate two-level logic minimization and PAC learning with membership queries. *J. Comput. System Sci.*, 75(1):13–26, 2009. Preliminary versions in STOC'06 and ECCC. [doi:10.1016/j.jcss.2008.07.007] 2

[13] ODED GOLDREICH, SHAFI GOLDWASSER, AND SILVIO MICALI: How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Preliminary version in FOCS'84. [doi:10.1145/6490.6503] 2

[14] ODED GOLDREICH, SILVIO MICALI, AND AVI WIGDERSON: Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991. Preliminary version in FOCS'86. [doi:10.1145/116825.116852] 2

[15] JOHAN HÅSTAD: Almost optimal lower bounds for small depth circuits. In *Proc. 18th STOC*, pp. 6–20. ACM Press, 1986. [doi:10.1145/12130.12132] 6

[16] SHUICHI HIRAHARA AND OSAMU WATANABE: Limits of minimum circuit size problem as oracle. In *Proc. 31st IEEE Conf. on Computational Complexity (CCC'16)*, volume 50, pp. 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. Available at ECCC. [doi:10.4230/LIPIcs.CCC.2016.18] 18

[17] JOHN M. HITCHCOCK AND ADURI PAVAN: On the NP-completeness of the minimum circuit size problem. In *35th IARCS Conf. Found. of Software Tech. and Theoret. Comput. Sci. (FSTTCS'15)*, volume 45 of *LIPIcs*, pp. 236–245, 2015. [doi:10.4230/LIPIcs.FSTTCS.2015.236] 18

[18] RUSSELL IMPAGLIAZZO, VALENTINE KABANETS, AND AVI WIGDERSON: In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. System Sci.*, 65(4):672–694, 2002. Preliminary version in CCC'01. [doi:10.1016/S0022-0000(02)00024-7] 6

[19] RUSSELL IMPAGLIAZZO AND AVI WIGDERSON: $\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. 29th STOC*, pp. 220–229. ACM Press, 1997. [doi:10.1145/258533.258590] 3, 15

[20] VALENTINE KABANETS AND JIN-YI CAI: Circuit minimization problem. In *Proc. 32nd STOC*, pp. 73–79. ACM Press, 2000. [doi:10.1145/335305.335314] 2, 5, 6, 8, 9, 14

[21] CODY D. MURRAY AND R. RYAN WILLIAMS: On the (non) NP-hardness of computing circuit complexity. In *Proc. 30th IEEE Conf. on Computational Complexity (CCC'15)*, volume 33 of *LIPIcs*, pp. 365–380. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. [doi:10.4230/LIPIcs.CCC.2015.365] 1

[22] CHRISTOS H. PAPADIMITRIOU AND MIHALIS YANNAKAKIS: A note on succinct representations of graphs. *Inf. Control*, 71(3):181–185, 1986. [doi:10.1016/S0019-9958(86)80009-2] 4

[23] SVEN SKYUM AND LESLIE G. VALIANT: A complexity theory based on Boolean algebra. *J. ACM*, 32(2):484–502, 1985. Preliminary version in SFCS'81. [doi:10.1145/3149.3158] 4

[24] N. VARIYAM VINODCHANDRAN: Nondeterministic circuit minimization problem and derandomizing Arthur-Merlin games. *Int. J. Found. Comput. Sci.*, 16(6):1297–1308, 2005. [doi:10.1142/S0129054105003819] 5

[25] HERIBERT VOLLMER: *Introduction to Circuit Complexity: A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999. [doi:10.1007/978-3-662-03927-4] 18

[26] R. RYAN WILLIAMS: Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. Preliminary version in STOC'13. [doi:10.1137/130938219, arXiv:1212.1891] 6, 8

[27] STANISLAV ŽÁK: A Turing machine time hierarchy. *Theoret. Comput. Sci.*, 26(3):327–333, 1983. [doi:10.1016/0304-3975(83)90015-4] 14

AUTHORS

Cody D. Murray
Ph. D. student
MIT, Cambridge, MA
cdmurray@mit.edu

R. Ryan Williams
Associate professor
MIT, Cambridge, MA
rrw@mit.edu
http://csail.mit.edu/~rrw/

ABOUT THE AUTHORS

CODY D. MURRAY got an undergraduate degree from UCSD in 2013, a Masters from Stanford in 2016, and is now a Ph. D. student at MIT. His advisor is Ryan Williams. At Stanford, he enjoyed riding his bike around Palo Alto, and he enjoys designing cool games. He doesn't like to talk about himself.

R. RYAN WILLIAMS was an assistant professor at Stanford from 2011 to 2016, and is now an associate professor at MIT. He got his Ph. D. from Carnegie Mellon in 2007, advised by Manuel Blum. He doesn't like to talk about himself, either.