Electronic Colloquium on Computational Complexity, Report No. 164 (2014)

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

Cody D. Murray
Stanford University
cdmurray@stanford.edu

Ryan Williams
Stanford University
rrw@cs.stanford.edu

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 poly-logarithmic time projections, we show that MCSP is *provably not* NP-hard under $O(n^{1/2-\varepsilon})$-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: NP $\not\subset$ P$_{/\text{poly}}$ and E $\not\subset$ i.o.-SIZE$(2^{\delta n})$ for some $\delta > 0$ (hence P = BPP also follows). We show that even the NP-hardness of MCSP under general polynomial-time reductions would separate complexity classes: EXP $\neq$ NP $\cap$ P$_{/\text{poly}}$, which implies EXP $\neq$ 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 Σ_2P-hardness of NMCSP, even under arbitrary polynomial-time reductions, would imply EXP $\not\subset$ P$_{/\text{poly}}$.

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 2^n bits (for some n), 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's 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 $O(k \log k) \leq O(|T|)$ time, then verified on all bits of the truth table T in $\text{poly}(2^n, k) \leq \text{poly}(|T|)$ time.[1]

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 [All14]. More notes on the history of this problem can be found in [KC00].)

Lower Bounds for MCSP? There is substantial evidence that MCSP $\notin$ P. The Natural Proofs work of Razborov and Rudich [RR97] shows that if MCSP $\in$ P, then (essentially by definition) there is a P-natural property useful against P$_{/\text{poly}}$; therefore, efficient algorithms for MCSP imply that there are no pseudorandom functions. Kabanets and Cai [KC00] made this critical observation, noting that the hardness

[1] Recall that every Boolean function $f : \{0,1\}^n \to \{0,1\}$ has a circuit of size at most $k \leq (1+o(1))2^n/n$ [Lup59]. Hence every instance $\langle T, k\rangle$ with $k > 2|T|/\log|T|$ is a yes-instance of MCSP.

ISSN 1433-8092

of factoring Blum integers implies that MCSP is hard. Allender *et al.* [ABK+06] 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 $\mathsf{ZPP}^{\mathrm{MCSP}}$. (Furthermore, [ABK+06] also prove that MCSP $\notin \mathsf{AC0}$.) Allender and Das [AD14] recently showed that Graph Isomorphism is in $\mathsf{RP}^{\mathrm{MCSP}}$, and in fact every problem with statistical zero-knowledge interactive proofs 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 [AHM+08]. Kabanets and Cai [KC00] show that, if MCSP is NP-complete under so-called "natural" poly-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 $\mathsf{NP} \subset \mathsf{SUBEXP}$. Therefore NP-completeness under a restricted reduction type would imply (expected) circuit lower bounds. Allender *et al.* [ABK+06] show that if $\mathsf{PH} \subset \mathsf{SIZE}(2^{n^{o(1)}})$ then MCSP is not hard for TC^0 under 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 [ABK+06], and recently Allender and Holden [AH14] 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 AC0 reductions, then $\mathsf{P} \neq \mathsf{NP}$.

NP-completeness has been defined for many different reducibility notions: polynomial time, logarithmic space, 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 1.1.)
2. Under slightly stronger reductions such as uniform AC0, the NP-completeness of MCSP would imply $\mathsf{NP} \not\subset \mathsf{P}_{/\mathrm{poly}}$[2] and $\mathsf{E} \not\subset$ i.o.-$\mathsf{SIZE}(2^{\delta n})$ for some $\delta > 0$, therefore $\mathsf{P} = \mathsf{BPP}$ as well by [IW97].
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 quite far from a proof at the present time.

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 output 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*

[2] After learning of our preliminary results, Eric Allender and Dhiraj Holden [AH14] independently established $\mathsf{P} \neq \mathsf{NP}$ as a consequence, and Valentine Kabanets (personal communication) found an alternative proof of the implication for $\mathsf{NP} \not\subset \mathsf{P}_{/\mathrm{poly}}$.

- $x \in L \iff R(x,1) \cdot R(x,2) \cdots R(x,\ell_x) \in L'$.

(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 "local" structure transforming small pieces of the input to small pieces of the output.[3] 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 ith bit of $R(x)$ for all x of length n, or a $j = 1,\ldots,n$ with $b \in \{0,1\}$ such that the ith bit of $R(x)$ (for all x of length n) equals $b \cdot x_j + (1-b) \cdot (1-x_j)$. Skyum and Valiant [SV85] observed that almost all NP-complete problems are also complete under projections. So for example, we have:

Proposition 1 ([SV85, PY86]) *SAT, Vertex Cover, Independent Set, Hamiltonian Path, and 3-Coloring are* NP*-complete under* $\mathsf{TIME}(\mathit{poly}(\log n))$ *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 1.1 *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.*[4]

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

Hardness Under Stronger Reducibilities. For stronger reducibility notions than sub-polynomial time, we do not yet have unconditional non-hardness results for MCSP. (Of course, a proof that MCSP is not NP-complete under poly-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 1.2 *If MCSP is* NP*-hard under polynomial-time reductions, then $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P}_{/poly}$. Consequently, $\mathsf{EXP} \neq \mathsf{ZPP}$.*

Corollary 1.1 *If MCSP is* NP*-hard under logspace reductions, then $\mathsf{PSPACE} \neq \mathsf{ZPP}$.*

Theorem 1.3 *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.*

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 [KC00]). We conjecture that the consequence of Theorem 1.2 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.

Σ_2-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" implies circuit lower bounds.

[3] 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 $\mathsf{AC0}$.

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

We have not yet shown a result like this (but come close with $\mathsf{EXP} \neq \mathsf{ZPP}$ in Theorem 1.2). 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 $\Sigma_2\mathsf{P}$: nondeterministically guess a circuit C with a "main" input and "auxiliary" input, nondeterministically evaluate C on all 2^n inputs x for which $T(x) = 1$, then universally verify on all 2^n 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 1.4 *If NMCSP is* MA*-hard under polynomial-time reductions, then* $\mathsf{EXP} \not\subset \mathsf{P}_{/poly}$.

Vinodchandran [Vin05] 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 new idea exploits 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. For the simplest example of the approach, let L be a unary (tally) language, and suppose we have a $\mathsf{TIME}(\mathrm{poly}(\log n))$ reduction R from L to MCSP. The outputs of R are pairs (T,k) where T is a truth table and k is the size parameter. Because each 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, and leads to many consequences.

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, [KC00, IKW02, Wil13] 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 [AB09]. 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 $O(s(n))$. Define $\Sigma_2\mathsf{TIME}[t(n)]$

to be the class of languages recognizable by a Σ_2 machine in time $O(t(n))$; more precisely, the languages L such that there exists a linear time machine M such that for all strings x,

$$x \in L \iff (\exists y \in \{0,1\}^{t(|x|)})(\forall z \in \{0,1\}^{t(|x|)})[M(x,y,z) \text{ accepts}].$$

In some of our results, we apply the well-known PARITY lower bound of Håstad:

Theorem 2.1 (Håstad [Hås86]) *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 sub-polynomial reductions. In Definition 1.1 we defined sub-polynomial 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 $y_1, \ldots, y_{2^k} \in \{0,1\}^k$ be the list of k-bit strings in lex order. Given $f : \{0,1\}^n \to \{0,1\}$, the truth table of f is defined to be $tt(f) := f(y_1) f(y_2) \cdots f(y_{2^n})$.

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's 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 *There are $\mathsf{TIME}(poly(\log n))$ 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 [All01, ABK+06]. Define MCSP′ to be the version with circuit size parameter set to $|T|^{1/2}$, that is, $MCSP' := \{T \mid CC(T) \leq |T|^{1/2}\}$. 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 2.1 ([Wil13]) *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 lex 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 either 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 1.1 *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.*

The proof 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 [KC00]) 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 $\tilde{O}(t(n))$, by exploiting naturalness. Then we use this bound on k to construct a Σ_2 algorithm for PARITY which existentially guesses an $\tilde{O}(t(n))$-size circuit for the truth table produced by the reduction, then universally verifies the circuit is correct on all bits of the truth table. Finally, we convert the Σ_2 algorithm into a depth-three circuit family of $2^{\tilde{O}(t(n))}$ size, and appeal to Håstad's 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 *Let $p(n)$ be a polynomial. There is an algorithm A which, 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 jth 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 $j = 1, \ldots, n + p$, A first 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, let $k = 0, \ldots, p - 1$ be such that $j < i_{k+1}$; then 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 3.1 (Kabanets-Cai [KC00]) *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 1 *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:

> $R'(x,i)$ 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 $|S| \leq t(2n)\log(2n)$. 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 $j \leq n + |S|$ then run the algorithm $A(x,S,j)$ from Proposition 3 to instead obtain the jth 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 $j \leq 2n$ then output 0, and if $j > 2n$ then output $\star$ (out of bounds). Since the algorithm of Proposition 3 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 string $2n$-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 $R'(z,i)$. 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 doesn't exceed n). As the instance f of MCSP has $CC(f) \leq s$, by Lemma 2.1 the truth table T in the instance $tt(f)$ has $CC(T) \leq cs$ 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 $\tilde{O}(t(n)\log^2 n)$. □

Now, we show that efficient reductions from PARITY to MCSP yield efficient Σ_2 algorithms for PARITY:

Claim 2 *If there is a* $\mathsf{TIME}(t(n))$ *reduction from PARITY to MCSP, then there is a* $\Sigma_2\mathsf{TIME}(\tilde{O}(t(n)))$ *algorithm for PARITY.*

Proof. Construct a Σ_2 algorithm for PARITY as follows:

> Given an input x, existentially guess a circuit C with $O(\log n)$ inputs and size at most $s = \tilde{O}(t(n))$, where s is taken from Claim 1. Then universally verify over all possible $O(\log n)$-bit inputs i to C that $C(i) = R'(x,i)$, where R' is from Claim 1. If yes, then *accept*, else *reject*.

Since we know the value of the size parameter in the instance output by $R'(x,\cdot)$ is at most s (from Claim 1), there is a circuit C of size at most s with the above property if and only if x has odd parity. Since the number of inputs to C is $O(\log n)$, the universal quantification in the above procedure is only $O(\log n)$ bits. Verification also takes $\tilde{O}(t(n))$ time, since C can be evaluated in $\tilde{O}(t(n))$ time on any input. Hence the Σ_2 procedure has the claimed running time. □

Finally, we can complete the proof of Theorem 1.1:

Proof of Theorem 1.1. Suppose that PARITY has a $\mathsf{TIME}(n^\delta)$ reduction from PARITY to MCSP, for some $\delta < 1/2$. Then by Claim 2, there is a Σ_2 algorithm for PARITY running in $\tilde{O}(n^\delta)$ time. Such an algorithm can be converted into a depth-three OR-AND-OR circuit of size $2^{\tilde{O}(n^\delta)}$: the top OR at the output has incoming wires for all possible $2^{\tilde{O}(n^\delta)}$ existential guesses for the Σ_2 machine, the middle AND tries all $2^{\tilde{O}(n^\delta)}$ universal guesses, and the remaining deterministic computation on $\tilde{O}(n^\delta)$ bits 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 AC0 circuits of size $2^{\tilde{O}(n^\delta)}$. For $\delta < 1/2$, this is false by Håstad (Theorem 2.1). □

Remark 1 *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 depth-three 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

Reminder of Theorem 1.2 *If MCSP is* NP*-hard under polynomial-time reductions, then* $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P}_{/poly}$. *Consequently,* $\mathsf{EXP} \neq \mathsf{ZPP}$.

Reminder of Corollary 1.1 *If MCSP is* NP*-hard under logspace 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* $\mathsf{EXP} \subseteq \mathsf{P}_{/poly} \implies \mathsf{EXP} = \mathsf{NEXP}$.

Proof. Suppose that MCSP is hard for sparse NP languages under polynomial-time reductions, and that $\mathsf{EXP} \subseteq \mathsf{P}_{/poly}$. 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

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

BITS_R is clearly in EXP. Since $\mathsf{EXP} \subseteq \mathsf{P}_{/poly}$, 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 $tt(D)$ is the output of $R(x)$. 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 2.1. This observation leads to the following exponential time algorithm for L:

> On input x, run the reduction $R(x)$, obtaining an exponential sized 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 sized 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 4.1 *If MCSP is* NP*-hard under logspace reductions, then* $\mathsf{PSPACE} \subseteq \mathsf{P}_{/poly} \implies \mathsf{NEXP} = \mathsf{PSPACE}$.

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 1.2, that NP-hardness of MCSP implies $\mathsf{EXP} \neq \mathsf{NP} \cap \mathsf{P}_{/\mathrm{poly}}$:

Proof of Theorem 1.2. 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 [Ž83]. □

Corollary 1.1 immediately follows from the same argument as Theorem 1.2, applying Corollary 4.1.

We would like to strengthen Theorem 1.2 to show that the NP-hardness of MCSP actually implies *circuit* lower bounds such as $\mathsf{EXP} \not\subset \mathsf{P}_{/\mathrm{poly}}$. 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 [KC00].)

4.2 Consequences of NP-Hardness under AC0 Reductions

Now we turn to showing consequences of assuming that MCSP is NP-hard under uniform AC0 reductions. Here we obtain consequences so strong that we are skeptical the hypothesis is true.

Reminder of Theorem 1.3 *If MCSP is* NP*-hard under logtime-uniform* 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 4.2 *If MCSP is* NP*-hard under LOGTIME-uniform* AC0 *reductions, then* $\mathsf{NP} \subseteq \mathsf{P}_{/poly} \implies \mathsf{NEXP} \subseteq \mathsf{P}_{/poly}$.

Proof. The proof is similar in spirit to that of Theorem 4.1. Suppose that MCSP is NP-hard under LOGTIME-uniform AC0 reductions, and that $\mathsf{NP} \subseteq \mathsf{P}_{/poly}$. Then $\Sigma_k\mathsf{P} \subseteq \mathsf{P}_{/poly}$ 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 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-sized instances of MCSP, for some constant k. In particular, $S(x,i)$ outputs the ith bit of the reduction R on input $x01^t$, and S can be implemented in $\Sigma_k\mathsf{P}$, and hence in $\mathsf{P}_{/\mathrm{poly}}$ 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 $\mathrm{poly}(|x|)$-size circuits. Therefore by Lemma 2.1, the truth table of the MCSP instance being output on x must have a $\mathrm{poly}(|x|)$-size circuit. We can then decide L in $\Sigma_{k+2}\mathsf{P}$ time: on an input x, existentially guess a circuit C of $\mathrm{poly}(|x|)$ 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 $\mathsf{NEXP} \subseteq \Sigma_{k+2}\mathsf{P} \subseteq \mathsf{P}_{/poly}$. □

Theorem 4.3 *If MCSP is* NP*-hard under* P*-uniform* AC0 *reductions, then* $\mathsf{E} \not\subset$ *i.o.-*$\mathsf{SIZE}(2^{\delta n})$ *for some* $\delta > 0$*. As a consequence,* $\mathsf{P} = \mathsf{BPP}$ *also follows from the assumption (Impagliazzo and Wigderson [IW97]).*

Proof. Assume the opposite: that MCSP is NP-hard under P-uniform AC0 reductions and for every $\varepsilon > 0$, $\mathsf{E} \subset$ i.o.-$\mathsf{SIZE}(2^{\varepsilon n})$. By Agrawal *et al.* [AAI+01] (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. By Claim 1, 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.

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, $i = 1,\ldots,n^c+c$, 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 $2^{O(m)} \leq n^{O(1)}$. By assumption, for every $\varepsilon > 0$, R' has circuits $\{D_m\}$ of size $O(2^{\varepsilon m}) \leq O(n^{2\varepsilon})$ 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 $O(n^{2\varepsilon})$-size circuit D_n, the circuit complexity of T_m is at most $O(n^{2\varepsilon})$, by Lemma 2.1. 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 $O(n^{2\varepsilon})$, the *same* circuit D_m of size $O(n^{2\varepsilon})$ 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 an i.o.-$\Sigma_2\mathsf{TIME}(\tilde{O}(n^{2\varepsilon}))$ algorithm A, as follows:

> On input (x,D), where $n=|x|$ and D is an $O(n^{2\varepsilon})$ 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 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.
>
> Next, nondeterministically guess a circuit C' of size at most k; we wish to verify that for all i, $C'(i)$ outputs the i^{th} bit of the truth table T_x produced by the NC0 reduction on input x. We can verify this by noting that the i^{th} bit of T_x can also be computed in $O(n^{\varepsilon})$ time, via D. Namely, universally choose an i, and produce the $O(1)$-size subcircuit that computes the i^{th} output bit of the 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', $A(x,D)$ computes PARITY correctly. For every $\varepsilon>0$, there are infinitely many m such that the circuit size parameter k is at most $2^{O(\varepsilon m)}\leq O(n^{2\varepsilon})$, and the circuit D of size $2^{O(\varepsilon m)}\leq O(n^{2\varepsilon})$ exists. Under these conditions, the above Σ_2 algorithm A runs in $\tilde{O}(n^{2\varepsilon})$ time. As a result, for every $\varepsilon>0$ we can find for infinitely many n such that the algorithm A has a corresponding depth-3 AC0 circuit of $2^{\tilde{O}(n^{\varepsilon})}$ size. Suppose we hardwire the $O(n^{2\varepsilon})$-size circuit D that computes R' into the corresponding AC0 circuit, on input lengths n for which D exists. Then for all $\varepsilon>0$, PARITY can be solved infinitely often with depth-3 AC0 circuits of $2^{\tilde{O}(n^{\varepsilon})}$ size, contradicting Håstad (Theorem 2.1). □

Proof of Theorem 1.3. Suppose MCSP is NP-hard under logtime-uniform AC0 reductions. The consequence $\mathsf{E}\not\subset\mathsf{SIZE}(2^{\delta n})$ was already established in Theorem 4.3. The consequence $\mathsf{NP}\not\subset\mathsf{P}_{/poly}$ follows immediately from combining Theorem 4.2 and Theorem 4.3. □

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 $x\in\{0,1\}^n$, $f(x)=1\iff$ there is a y of length at most s 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 $s\cdot\mathrm{poly}(\log s)$. Therefore we 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 Σ_2P: 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 Σ_2P-hard. (The proof of Theorem 1.4 below will work for the depth-two version and the unrestricted version all the same.)

Recall that it is *known* that MCSP for depth-two circuits is NP-hard [AHM+08]. That is, the "deterministic counterpart" of MCSP is known to be NP-hard.

Reminder of Theorem 1.4 *If NMCSP is* MA*-hard under polynomial-time reductions, then* $\mathsf{EXP}\not\subseteq\mathsf{P}_{/poly}$.

Proof. Suppose that NMCSP is MA-hard under polynomial-time reductions, and suppose that EXP $\subseteq$ $\mathsf{P}_{/poly}$. 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}-sized instances of NMCSP with a size parameter $s(x)$ on input x. Since R' runs in exponential time, and we assume EXP is in $\mathsf{P}_{/poly}$, there is a k such that for all x, there is a nondeterministic circuit $C((x,i),y)$ of $\leq n^k + k$ size that computes the ith bit of $R'(x)$. Therefore we know that $s(x) \leq |x|^k + k$ 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 $s(x) \leq |x|^k + k$, 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 $\mathsf{MAEXP} = \mathsf{EXP} \subseteq \mathsf{P}_{/\mathrm{poly}}$, which contradicts the circuit lower bound of Buhrman, Fortnow, and Thierauf [BFT98]. □

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 EXP $\not\subset \mathsf{P}_{/\mathrm{poly}}$. We showed that if MCSP is hard for sparse NP languages then EXP $\neq$ ZPP; surely a reduction from SAT to MCSP would provide a stronger consequence.
- MCSP is (unconditionally) not NP-hard under logtime-uniform AC0 reductions. Theorem 1.1 already implies that MCSP isn't NP-hard under polylogtime-uniform NC0 reductions. Perhaps this next step isn't far away, since we already know that hardness under P-uniform AC0 reductions implies hardness under P-uniform NC0 reductions (by Agrawal *et al.* [AAI+01]).

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 [AHM+08]. Since there are $2^{\Omega(n^\delta)}$ size lower bounds against AC0, can it be proved that finding the minimum AC0 circuit for a given truth table is QuasiNP-hard? In general, can circuit lower bounds imply hardness results for circuit minimization?

Acknowledgements. We thank Greg Bodwin and Brynmor Chapman for discussions on these results. We also thank Eric Allender for providing a preprint of his work with Dhiraj Holden.

References

[AAI+01] Manindra Agrawal, Eric Allender, Russell Impagliazzo, Toniann Pitassi, and Steven Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10(2):117–138, 2001.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[AD14] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Mathematical Foundations of Computer Science (MFCS), Part II*, pages 25–32, 2014.

[AH14] Eric Allender and Dhiraj Holden. The minimum oracle circuit size problem. Manuscript, submitted for publication, 2014.

[AHM+08] 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.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *FSTTCS*, volume 2245 of *LNCS*, pages 1–15. Springer, 2001.

[All14] Eric Allender. Personal communication, 2014.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *CCC*, pages 8–12, 1998.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20, 1986.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *JCSS*, 65(4):672–694, 2002.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000.

[Lup59] O. B. Lupanov. A method of circuit synthesis. *Izvestiya VUZ, Radiofizika*, 1(1):120–140, 1959.

[PY86] Christos H. Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71(3):181–185, 1986.

[RR97] Alexander Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[SV85] Sven Skyum and Leslie G Valiant. A complexity theory based on boolean algebra. *J. ACM*, 32(2):484–502, 1985.

[Vin05] N. V. Vinodchandran. Nondeterministic circuit minimization problem and derandomizing Arthur-Merlin games. *International Journal of Foundations of Computer Science*, 16(6):1297–1308, 2005.

[Wil13] Ryan Williams. Natural proofs versus derandomization. In *STOC*, pages 21–30, 2013.

[Žák83] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, 1983.

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 WLOG that $k \leq 2|T|/\log|T|$: the size of a minimum circuit for a boolean function on n inputs is always less than $2^{n+1}/n$. 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 $\mathsf{TIME}(\text{poly}(\log n))$ reductions to and from this encoding to the others above, mainly because k' has length $O(\log n)$.

Proposition 4 *There are* $\mathsf{TIME}(poly(\log n))$ *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 don't 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 $\text{poly}(\log n)$ 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 $O(\log|y|)$ calls to the reduction.) From here, outputting the ith bit of either T or k in the binary encoding is easy, since $|k| = O(\log|y|)$.

□

ECCC ISSN 1433-8092

http://eccc.hpi-web.de