The minimum oracle circuit size problem

Citation to Publisher Version: Allender, Eric, Holden, Dhiraj & Kabanets, Valentine. (2016). The minimum oracle circuit size problem. *Computational Complexity 26*(2), 469-496.http://dx.doi.org/10.1007/s00037-016-0124-0.

Citation to *this* Version: Allender, Eric, Holden, Dhiraj & Kabanets, Valentine. (2016). The minimum oracle circuit size problem. *Computational Complexity 26*(2), 469-496. Retrieved from doi:10.7282/T31G0PDW.

THE MINIMUM ORACLE CIRCUIT SIZE PROBLEM

Eric Allender, Dhiraj Holden,
and Valentine Kabanets

January 28, 2016

Abstract. We consider variants of the Minimum Circuit Size Problem MCSP, where the goal is to minimize the size of *oracle* circuits computing a given function. When the oracle is QBF, the resulting problem $\mathsf{MCSP}^{\mathsf{QBF}}$ is known to be complete for PSPACE under ZPP reductions. We show that it is *not* complete under logspace reductions, and indeed it is not even hard for TC^0 under uniform AC^0 reductions. We obtain a variety of consequences that follow if oracle versions of MCSP are hard for various complexity classes under different types of reductions. We also prove analogous results for the problem of determining the resource-bounded Kolmogorov complexity of strings, for certain types of Kolmogorov complexity measures.

Keywords. Kolmogorov complexity, Minimum Circuit Size Problem, PSPACE, NP-Intermediate sets

Subject classification. F.1.3 Complexity Measures and Classes

1. Introduction

The Minimum Circuit Size Problem (MCSP) asks to decide, for a given truth table f of a Boolean function and a parameter s, whether f is computable by a Boolean circuit of size at most s. MCSP is a well-known example of a problem in NP that is widely believed to be intractable, although it is not known to be NP-complete. MCSP is known to be hard for the complexity class SZK under BPP-Turing reductions (Allender & Das 2014), which provides strong evidence for intractability. On the other hand, Kabanets & Cai (2000) showed that if MCSP is NP-complete under the "usual" sort of polynomial-time reductions, then $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$. This can not be interpreted as strong evidence against NP-completeness – since it is widely conjectured that $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$ – but it does indicate that it may be difficult to provide an NP-completeness proof.

However, there are other ways to define what the "usual" sort of reductions are: e.g., logspace, (uniform) TC^0, AC^0, or NC^0. The overwhelming majority of problems that are known to be NP-complete are, in fact, NP-complete under very restricted kinds of reductions. Can we rule out NP-hardness of MCSP under such reductions?

Very recently, Murray & Williams (2015) have shown that MCSP is not even P-hard under uniform NC^0 reductions. Can MCSP be NP-hard under slightly stronger reductions, e.g., uniform AC^0 reductions? We suspect that the answer is 'No', but so far we (like Murray and Williams) can only show that P-hardness of MCSP under uniform AC^0, TC^0, or logspace reductions would imply new (likely) complexity lower bounds (in the spirit of (Kabanets & Cai 2000)).

The main focus of the present paper is an *oracle version* of MCSP, denoted MCSP^A for a language A, which asks to decide for a given truth table f and a parameter s if f is computable by an A-oracle circuit of size at most s. We prove a number of implications of hardness of MCSP^A for various choices of the oracle A, and various reductions. In particular, we prove for a PSPACE-complete A that MCSP^A is *not* P-hard under uniform AC^0 reductions. (See Theorem 3.9.)

The results presented here (along with the results recently reported by Murray & Williams (2015)) are the first results giving unlikely consequences that would follow if variants of MCSP or the various oracle circuit minimization problems are hard under a natural notion of reducibility. We also show that analogous results hold in the Kolmogorov complexity setting due to the correspondence between circuit size and Kolmogorov complexity, using the minimum-KT complexity problem defined in this paper.

Below we provide a summary of our main results.

1.1. Our results. Most of our results follow the template:

> *If* $\mathsf{MCSP}^{\mathsf{A}}$ *is hard for a complexity class* $\mathcal{C}$ *under reductions of type* $\mathcal{R}$*, then complexity statement* $\mathcal{S}$ *is true.*

Table 1.1 below states our results for different instantiations of A, $\mathcal{C}$, $\mathcal{R}$, and $\mathcal{S}$; note that $\mathcal{S} = \bot$ means that the assumption is false, i.e., $\mathsf{MCSP}^{\mathsf{A}}$ is *not* $\mathcal{C}$-hard under $\mathcal{R}$-reductions. Throughout, we assume that the reader is familiar with complexity classes such as NP, PP, PSPACE, NEXP, etc. We denote the polynomial hierarchy by PH, and its linear-time version (linear-time hierarchy) by LTH. The Counting Hierarchy, denoted CH, is the union of the classes $\mathsf{PP}, \mathsf{PP}^{\mathsf{PP}}$, etc. CH was defined by Torán (1991) and arises frequently in complexity investigations. LTH was studied by Wrathall (1978), who showed that it coincides with the Rudimentary sets of Smullyan (1961). LTH has been studied, for instance, in (Allender & Gore 1991, 1993; Fortnow 2000; McKenzie *et al.* 2010).

Table 1.1: Summary of main results: If $\mathsf{MCSP}^{\mathsf{A}}$ is $\mathcal{C}$-hard under $\mathcal{R}$, then $\mathcal{S}$. The last column shows the theorem where the result is stated in the paper.

oracle A	class $\mathcal{C}$	reductions $\mathcal{R}$	statement $\mathcal{S}$	Theorem
PH-hard	TC^0	uniform AC^0	$\bot$	Theorem 3.9
any	TC^0	uniform AC^0	$\mathsf{LTH} \not\subseteq \mathsf{io{-}SIZE}^A[2^{\Omega(n)}]$	Lemma 3.10
any	TC^0	uniform AC^0	$\mathsf{NP}^A \not\subseteq \mathsf{SIZE}^A[\mathsf{poly}]$	Corollary 3.13
any in CH	P	uniform TC^0	$\mathsf{P} \neq \mathsf{PP}$	Corollary 3.2
$\emptyset$	P	logspace	$\mathsf{P} \neq \mathsf{PSPACE}$	Corollary 3.3
QBF	P	logspace	$\mathsf{EXP} = \mathsf{PSPACE}$	Corollary 3.7
QBF	NP	logspace	$\mathsf{NEXP} = \mathsf{PSPACE}$	Theorem 3.6
QBF	PSPACE	logspace	$\bot$	Corollary 3.8
EXP-complete	NP	polytime	$\mathsf{NEXP} = \mathsf{EXP}$	Theorem 3.4

For the most restricted reductions, uniform AC^0, we get that $\mathsf{MCSP}^{\mathsf{A}}$ is not TC^0-hard for any oracle A such that $\mathsf{PH} \subseteq \mathsf{SIZE}^A[\mathsf{poly}]$ (Theorem 3.9), e.g., for $A = \oplus\mathsf{P}$ (Corollary 3.12). For any oracle A, we conclude new circuit lower bounds for the linear-time hierarchy and for NP^A (Lemma 3.10 and Corollary 3.13[1]).

[1]Prior to our work, Murray & Williams (2015) have shown that if $\mathsf{SAT} \leq_m^{\mathsf{AC}^0} \mathsf{MCSP}$, then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$. Their result is similar to (and is implied by) our Corollary 3.13 for the case of $A = \emptyset$.

If MCSP is P-hard under uniform TC^0 or logspace reductions, then P is different from PP or from PSPACE (Corollary 3.2 and Corollary 3.3).

One of the more interesting oracle circuit minimization problems is $\mathsf{MCSP}^{\mathsf{QBF}}$. It was shown by Allender *et al.* (2006) that $\mathsf{MCSP}^{\mathsf{QBF}}$ is complete for PSPACE under ZPP-Turing reductions, but the question of whether it is complete for PSPACE under more restrictive reductions was left open. For most natural complexity classes $\mathcal{C}$ above PSPACE, there is a corresponding oracle circuit minimization problem (which we will sometimes denote $\mathsf{MCSP}^{\mathcal{C}}$) that is known to be complete under P/poly reductions, but is not known to be complete under more restrictive reductions (Allender *et al.* 2006). For the particular case of $\mathcal{C} = \mathsf{PSPACE}$, we denote this as $\mathsf{MCSP}^{\mathsf{QBF}}$. We show that $\mathsf{MCSP}^{\mathsf{QBF}}$ is not PSPACE-complete under logspace reductions (Corollary 3.8). Furthermore, it is not even TC^0-hard under uniform AC^0 reductions (Theorem 3.9).

Finally, for even more powerful oracles A, we handle even general polynomial-time reductions. We show that if $\mathsf{SAT} \leq^p_m \mathsf{MCSP}^{\mathsf{EXP}}$, then $\mathsf{EXP} = \mathsf{NEXP}$ (Theorem 3.4).

We believe that MCSP is not TC^0-hard under even *nonuniform* AC^0 reductions. While we are unable to prove this, we can rule out restricted AC^0 reductions for a certain gap version of MCSP. Define *gap*-MCSP as follows: Given a truth table f and a parameter s, output 'No' if f requires circuit size s, and output 'Yes' if f can be computed by a circuit of size at most $s/2$. Call a mapping from n-bit strings to m-bit strings $\alpha(n)$-*stretching* if $m \leq n \cdot \alpha(n)$, for some function $\alpha : \mathbb{N} \to \mathbb{R}^{\geq 0}$.

We prove that gap-MCSP is *not* TC^0-hard under nonuniform AC^0 reductions that are $n^{1/31}$-stretching (Theorem 3.17).

1.2. Related work. The most closely related is the recent paper by Murray & Williams (2015), which also considers the question whether MCSP is NP-complete under weak reductions, and proves a number of conditional and unconditional results. The main unconditional result is that MCSP is *not* TC^0-hard under uniform NC^0 reductions (or more generally, under $O(n^{1/2-\epsilon})$-time projections, for every $\epsilon > 0$); we give an alternative proof of this result (Theorem 3.15). For conditional results, Murray & Williams (2015) show that if MCSP is NP-hard under uniform AC^0 reductions, then $\mathsf{NP} \not\subset \mathsf{P/poly}$ and $\mathsf{E} \not\subset \mathsf{io{-}SIZE}[2^{\Omega(n)}]$ (also implied by our Corollary 3.13 and Lemma 3.10), and that NP-hardness of MCSP under general polynomial-time reductions implies $\mathsf{EXP} \neq \mathsf{ZPP}$.

MCSP, $\mathsf{MCSP}^{\mathsf{QBF}}$ and other oracle circuit minimization problems are closely related to notions of resource-bounded Kolmogorov complexity. Briefly, a small (oracle) circuit is a short description of the string that represents the truth-table of the function computed by the circuit. Notions of resource-bounded Kolmogorov complexity that are roughly equivalent to (oracle) circuit size were presented and investigated by Allender *et al.* (2006).

In particular, there is a space-bounded notion of Kolmogorov complexity, KS, such that the set of KS-random strings (denoted R_{KS}) is complete for PSPACE under ZPP reductions. It is shown in (Allender *et al.* 2006) that R_{KS} is not even hard for TC^0 under AC^0 reductions, and R_{KS} is not hard for PSPACE under logspace-Turing reductions. The proof of this non-hardness result also carries over to show that a set such as $\{f : f$ is the truth table of a function on n variables that has QBF circuits of size at most $2^{n/2}\}$ is also not hard for TC^0 under AC^0 reductions, and is not hard for PSPACE under logspace-Turing reductions. However it does *not* immediately carry over to $\mathsf{MCSP}^{\mathsf{QBF}}$, which is defined as $\{(f, i) : f$ is the truth table of a function on n variables that has QBF circuits of size at most $i\}$; similarly it does not carry over to the set $\{(x, i) : \mathsf{KS}(x) \leq i\}$. Also, the techniques presented in (Allender *et al.* 2006) have not seemed to provide any tools to derive consequences assuming completeness results for oracle circuit minimization problems for oracles less powerful than PSPACE. We should point out, however, that (Allender *et al.* 2006) proves a result similar to (and weaker than) our Lemma 3.10 in the context of time-bounded Kolmogorov complexity: if R_{KT} is TC^0-hard under AC^0 many-one reductions, then $\mathsf{PH} \not\subseteq \mathsf{SIZE}\left[2^{n^{o(1)}}\right]$.

1.3. Our techniques. To illustrate our proof techniques, let us sketch a proof of one of our results: If MCSP is P-hard under uniform logspace reductions, then $\mathsf{P} \neq \mathsf{PSPACE}$ (Corollary 3.3).

The proof is by contradiction. Suppose that $\mathsf{P} = \mathsf{PSPACE}$. Our logspace reduction maps n-bit instances of QBF to n^c-bit instances (f, s) of MCSP so that each bit of f is computable in $O(\log n)$ space.

1. Imagine that our reduction is given as input a *succinct* version of QBF, where some $\mathsf{poly}(\log n)$-size circuit D on each $\log n$-bit input $1 \leq i \leq n$ computes the ith bit of the QBF instance. It is not hard to see that our reduction, given the circuit D, can compute each bit of f in $\mathsf{poly}(\log n)$ space. Thus the Boolean function with the truth table f is computable by a $\mathsf{PSPACE} = \mathsf{P}$ algorithm (which also has the circuit D as an input). It follows that this function f is computable by some polynomial-size Boolean circuit.

2. Next, since we know that f has at most polynomial circuit complexity, to decide the MCSP instance (f, s), we only need to consider the case where $s < \mathsf{poly}$ (since for big values of s, the answer is 'Yes'). But deciding such MCSP instances (which we call *succinct* MCSP) is possible in Σ_2^p: guess a circuit of size at most s, and verify that it agrees with the given polynomial-size circuit for f on all inputs.

3. Finally, since $\Sigma_2^p \subseteq \mathsf{PSPACE} = \mathsf{P}$, we get that our succinct MCSP instances can be decided in P. The reduction from succinct QBF to succinct MCSP is also in $\mathsf{PSPACE} = \mathsf{P}$. Hence, succinct QBF is in P. But, succinct QBF is $\mathsf{EXPSPACE}$-complete, and so we get the collapse $\mathsf{EXPSPACE} = \mathsf{P}$, contradicting the hierarchy theorems.

In step (1) of the sketched proof, the uniformity of an assumed reduction to MCSP is used to argue that the truth table f produced by the reduction is in fact "easy" to compute uniformly. The uniform complexity of computing the function f is roughly the "exponential" analogue of the uniform complexity of the reduction. For circuit classes such as AC^0 and TC^0, we use the well-known connection between the "exponential" analog of uniform AC^0 and PH, and between the "exponential" analog of uniform TC^0 and CH.

We use the uniform easiness of the function f to conclude that f has small circuit complexity (and hence our reduction actually outputs instances of *succinct* MCSP). To get that conclusion, we need to assume (or derive) the collapse to P/poly of the uniform complexity class that contains f; in our example above, we got it from the assumption that PSPACE = P.

Step (2) exploits the fact that succinct MCSP does *not* become "exponentially harder" (unlike the usual succinct versions of hard problems), but is actually computable in Σ_2^p.

In Step (3), we combine the algorithm for our reduction and the algorithm for succinct MCSP to get an "efficient" algorithm for the succinct version of the input problem (succinct QBF in our example). Since the succinct version of the input problem *does* become exponentially harder than its non-succinct counterpart, we get some impossible collapse (which can be disproved by diagonalization).

We use this style of proof for all our results involving reductions computable by uniform TC^0 and above. However, for the case of uniform AC^0 (and below), we get stronger results by replacing the diagonalization argument of Step (3) with the nonuniform AC^0 circuit lower bound for PARITY (Håstad 1989).

Remainder of the paper. We state the necessary definitions and auxiliary results in Section 2. Our main results are proved in Section 3, and some generalizations are given in Section 4. We give concluding remarks in Section 5.

2. Definitions

DEFINITION 2.1. *The minimum circuit size problem* MCSP, *as defined in (Kabanets & Cai 2000), is*

$$\{(f, s) \mid f \text{ has circuits of size } s\},$$

where f is a string of length 2^m encoding the entire truth-table of some m-variate Boolean function. (Versions of this problem have been studied long prior to (Kabanets & Cai 2000). See (Allender & Das 2014; Trakhtenbrot 1984) for a discussion of this history.) We will also consider the analogous problem for circuits with oracles, the Minimum A-Circuit Size problem MCSP^A, *defined analogously,* *where instead of ordinary circuits, we use circuits that also have oracle gates that query the oracle A. When A is a standard complete problem for some complexity class* $\mathcal{C}$*, we may refer to this as* $\mathsf{MCSP}^{\mathcal{C}}$*. Note that, for any oracle A,* $\mathsf{MCSP}^{\mathsf{A}} \in \mathsf{NP}^{\mathsf{A}}$.

We will not need to be very specific about the precise definition of the "size" of a circuit. Our results hold if the "size" of a circuit is the number of gates (including oracle gates), or the number of "wires", or the number of bits used to describe a circuit in some standard encoding. It is perhaps worth mentioning that the different versions of MCSP that one obtains using these different notions of "size" are not known to be efficiently reducible to each other.

Circuit size relative to oracle A is polynomially-related to a version of time-bounded Kolmogorov complexity, denoted KT^A, which was defined and studied by Allender *et al.* (2006).

DEFINITION 2.2. $KT^A(x) = min\{|d| + t \mid \forall b \in \{0,1,*\} \forall i \le |x| + 1\ U^A(d,i,b)$ *runs for at most t steps, and accepts iff* $x_i = b\}$. *Here, U is some fixed universal Turing machine, which has random access to the oracle A and to the input string (or "description") d;* x_i *denotes the i-th symbol of x, where* $x_{|x|+1} = *$.

By analogy to $\mathsf{MCSP}^{\mathsf{A}}$, we define the "minimum KT problem":

DEFINITION 2.3. $\mathsf{MKTP}^A = \{(x,i) \mid KT^A(x) \le i\}$.

All of our results that deal with $\mathsf{MCSP}^{\mathsf{A}}$ also apply to MKTP^A.

We wish to warn the reader that one's intuition can be a poor guide, when judging how $\mathsf{MCSP}^{\mathsf{A}}$ and $\mathsf{MCSP}^{\mathsf{B}}$ compare to each other, for given oracles A and B. For instance, it is known that $\mathsf{MCSP}^{\mathsf{SAT}}$ ZPP-Turing reduces to $\mathsf{MCSP}^{\mathsf{QBF}}$ (Allender *et al.* 2006), but no deterministic reduction is known. Similarly, no efficient reduction of any sort is known between MCSP and $\mathsf{MCSP}^{\mathsf{SAT}}$. Some of our theorems derive consequences from the assumption that $\mathsf{MCSP}^{\mathsf{SAT}}$ is hard for some complexity class under AC^0 reductions. Although one might suspect that this is a weaker hypothesis than assuming that MCSP is hard for the same complexity class under AC^0 reductions – certainly the best upper bound for $\mathsf{MCSP}^{\mathsf{SAT}}$ is worse than the best known upper bound for MCSP – nonetheless we are not able to derive the same consequences assuming only that MCSP is hard. For essentially all time- and space-bounded complexity classes $\mathcal{C}$ that contain PSPACE, $\mathsf{MCSP}^{\mathcal{C}}$ is complete for $\mathcal{C}$/poly under P/poly reductions (Allender *et al.* 2006, 2010), but uniform reductions are known only for two cases (Allender *et al.* 2006): when $\mathcal{C} = \mathsf{PSPACE}$ ($\mathsf{MCSP}^{\mathsf{QBF}}$ is complete for PSPACE under ZPP reductions) and when $\mathcal{C} = \mathsf{EXP}$ ($\mathsf{MCSP}^{\mathsf{EXP}}$ is complete for EXP under NP-Turing reductions).

2.1. Succinct Problems. The study of succinct encodings of computational problems was introduced by Galperin & Wigderson (1983); Papadimitriou & Yannakakis (1986), and has been studied since then by Balcázar *et al.* (1992); Wagner (1986), among others. Succinct encodings play an important role in the proofs of our main results.

DEFINITION 2.4. *Given a language L, we define the succinct version of L (denoted* succ.L*) to be the language* $\{C \mid tt(C) \in L\}$ *where C is a Boolean Circuit and $tt(C)$ is the truth-table for C.*

It will be necessary for us to consider “succinctly-presented” problems, where the circuit that constitutes the succinct description is itself an *oracle* circuit:

DEFINITION 2.5. *Given a language L and an oracle A, we define the A-succinct version of L (denoted A-*succ.L*) to be the language* $\{C \mid tt(C) \in L\}$ *where C is a Boolean Circuit with oracle gates, and $tt(C)$ is the truth-table for C, when it is evaluated with oracle A. If $A = \emptyset$, we denote this language as* succ.L.

The typical situation that arises is that the succinct version of a problem A has exponentially greater complexity than A. In particular, this happens when A is complete for a complexity class under “logtime reductions”.

DEFINITION 2.6. *We say that a function f can be computed in logarithmic time if there exists a random-access Turing machine that, given (x, i), computes the ith bit $f(x)$ in time $O(\log |x|)$.*

Building on prior work of (Galperin & Wigderson 1983; Papadimitriou & Yannakakis 1986; Wagner 1986), Balcázar *et al.* (1992) presented a large list of complexity classes ($\mathcal{C}_1$, $\mathcal{C}_2$), where $\mathcal{C}_1$ is defined in terms of some resource bound $B(n)$ and $\mathcal{C}_2$ is defined in the same way, with resource bound $B(2^n)$, such that if a set A is complete for $\mathcal{C}_1$ under logtime reductions, then succ.A is complete for $\mathcal{C}_2$ under polynomial-time many-one reductions. In particular, we will make use of the fact that succ.SAT is complete for NEXP, succ.MajSAT is complete for probabilistic exponential time, succ.QBF is complete for EXPSPACE, and succ.CVP is complete for EXP.

Somewhat surprisingly, the complexity of succ.MCSP appears *not* to be exponentially greater than that of MCSP. (Related observations were made earlier by Williams (2012).)

THEOREM 2.7. *succ.*MCSP $\in \Sigma_2^p$

PROOF. We present an algorithm in Σ_2^p that decides succ.MCSP. Given an instance of succinct MCSP C, note that $C \in$ succ.MCSP iff z is a string of the form $(f, s) \in$ MCSP, where $z = tt(C)$. By definition, $|z|$ must be a power of 2, say $|z| = 2^r$, and $|f|$ must also be a power of 2, say $|f| = 2^m$ for some $m < r$. Note also that if $s > |f| = 2^m$, then (f, s) should obviously be accepted, since every m-variate Boolean function has a circuit of size 2^m. To be precise, we will choose one particular convention for encoding the pair (f, s); other reasonable conventions will also yield a Σ_2^p upper bound. Let us encode (f, s) as a string of length 2^{m+1}, where the first 2^m bits give the truth table for f, and the second 2^m bits give s in binary. Note that this means that C has $m + 1$ input variables, and hardwiring the high-order input bit of C to 0 results in a circuit C' for f (of size at most $|C|$).

Using this encoding, the "interesting" instances (f, s) are of the form where the second half of the string is all zeros, except possibly for the low-order m bits (encoding a number $s \leq 2^m = |f|$). The low-order m bits can be computed deterministically in polynomial time, given C, by evaluating C on inputs $1^{m+1-\log m}0^{\log m}$, $1^{m+1-\log m}0^{-1+\log m}1, \ldots, 1^{m+1}$. Let the number encoded by the low-order m bits be s'. Then C (an encoding of (f, s)) is in succ.MCSP iff

- there is some bit position j corresponding to one of the high-order $2^m - m$ bits of s such that $C(j) = 1$, or
- there exists a circuit D of size at most s' such that, for all $i, D(i) = C'(i)$. (Note that $s' \leq s$.)

It is easily seen that this can be checked in Σ_2^p. □

Because this proof relativizes, we obtain:

COROLLARY 2.8. *Let A and B be oracles such that $B \leq_T^p A$. Then B-succ.MCSP^A is in $(\Sigma_2^p)^A$.*

PROOF. We use the same encoding as in Theorem 2.7. Thus, an oracle circuit C encoding an instance (f, s) (where f is an m-ary function) has $m+1$ input variables, and hardwiring the high-order input bit of C to 0 results in an oracle circuit C' (with oracle B) for f (of size at most $|C|$). But if $B \leq_T^p A$, then this also gives us an oracle circuit C'' (with oracle A) for f (of size at most $|C|^k$ for some k), where we can obtain C'' from C in polynomial time.

Then C (an encoding of (f, s)) is in B-succ.MCSP^A iff

- there is some bit position j corresponding to one of the high-order $2^m - m$ bits of s such that $C^B(j) = 1$, or
- there exists a circuit D of size at most s' such that, for all $i, D^A(i) = C''^A(i)$.

It is easily seen that this can be checked in $(\Sigma_2^p)^A$. □

An analogous result also holds for MKTP^A.

THEOREM 2.9. *Let A and B be oracles such that $B \leq_T^p A$. Then B-succ.MKTP^A is in $(\Sigma_2^p)^A$.*

PROOF. Given an instance of B-succ.MKTP^A C, note that $C \in B$-succ.MKTP^A only if z is a string of the form (x, i), where $z = tt(C)$. Let us settle on a suitable encoding for pairs; the number i should be at most $2|x|$ (a generous overestimate of how large

$\mathrm{KT}^A(x)$ could be), and thus should consist of at $O(\log |x|)$ bits. In order to mark the location of the "comma" separating x and i, we use the familiar convention of doubling each bit of i, and using the symbols 10 to mark the position of the "comma". Thus, given a circuit C with n variables, a $(\Sigma_2^p)^B$ machine can compute the length of the encoded string x as follows:

1. Using nondeterminism, guess a position ℓ and verify that $C^B(\ell) = 1$ and $C^B(\ell+1) = 0$.

2. Using co-nondeterminism, verify that for all $\ell' > \ell$ it is *not* the case that $C^B(\ell') = 1$ and $C^B(\ell'+1) = 0$. (If this test passes, then the $tt(C)$ is of the form (x, i) for some x and i, although it allows the possibility that absurdly large numbers i are provided.)

3. Reject if the number of bits used in the encoding of i is more than $4n$ (which is greater than $4 \log |x|$).

This $(\Sigma_2^p)^B$ computation can be simulated by a $(\Sigma_2^p)^A$, by our assumption that $B \leq_T^p A$.

The rest of the algorithm follows closely the algorithm that we presented for MCSP. Given a circuit C, guess the number ℓ such that $tt(C) = (x, i)$ for some string x of length ℓ. Without loss of generality, we can assume that the universal Turing machine used to define KT takes a description of a program and the input to the program and runs the program on that input. The universal oracle machine U, given a description d of length $|C| + |\ell| + O(1)$, along with (i, b) can output $*$ if $i > \ell$ and otherwise can use oracle A to simulate $C^B(i)$ and accept iff the answer is b. The running time will be at most $(|C| + |\ell|)^k$ for some k, which gives us an upper bound on $\mathrm{KT}^A(x)$.

The $(\Sigma_2^p)^A$ algorithm for B-succ.MKTP^A is thus:

1. Guess and verify ℓ as above, and in parallel:

2. Evaluate $C^B(j)$ for the $4n$ largest positions $j < 2^n$ (using oracle A), and thus obtain the encoding of i.

3. Accept if $i \geq (|C| + |\ell|)^k$.

4. Guess a description d' of length at most i. Reject if $U^A(d, \ell + 1, *)$ does not accept.

5. Using co-nondeterminism, verify that for all $j \leq \ell$ and all $b \in \{0, 1\}$, $U^A(d, j, b)$ accepts iff $C^B(j) = b$.

□

2.2. Constant-Depth Reductions.

PROPOSITION 2.10. *Suppose that f is a uniform AC^0 reduction from a problem A to a problem B. Let C be an instance of succ.A. Then, the language*

$$\{(C, i) \mid \text{the } i\text{th bit of } f(tt(C)) \text{ is } 1\}$$

is in LTH *(the linear-time hierarchy).*

PROOF. Consider the unary version of the above language:

$$\{1^{(C,i)} \mid \text{the } i\text{th bit of } f(tt(C)) \text{ is } 1\};$$

we claim that this language is in uniform AC^0. To see this, note that after computing the length of the input (in binary), and thus obtaining a description of C (of length $\log n$), an AC^0 algorithm can compute each bit of $tt(C)$. For instance, the ith bit of $tt(C)$ can be computed by guessing a bit vector of length $\log n$ recording the value of each gate of C on input i, and then verifying that all of the guessed values are consistent. Once the bits of $tt(C)$ are available, then the AC^0 algorithm computes $f(tt(C))$.

The result is now immediate, from (Allender & Gore 1993, Proposition 5), which shows that the rudimentary languages (that is, the languages in the linear-time version LTH of the polynomial-time hierarchy PH) are precisely the sets whose unary encodings are in Dlogtime-uniform AC^0. □

By an entirely analogous argument, we obtain:

PROPOSITION 2.11. *Suppose that f is a uniform TC^0 reduction from a problem A to a problem B. Let C be an instance of succ.A. Then, the language*

$$\{(C, i) \mid \text{the } i\text{th bit of } f(tt(C)) \text{ is } 1\}$$

is in CH.

3. Main Results

3.1. Conditional collapses and separations of complexity classes. Our first theorem shows that significant conclusions follow if MCSP is hard for P under AC^0 reductions. (Note that a stronger result appears later in the paper, as Corollary 3.13.)

THEOREM 3.1. *If there is any set A in the polynomial hierarchy such that $\mathsf{MCSP}^{\mathsf{A}}$ (or MKTP^A) is hard for P under uniform AC^0 reductions, then $\mathsf{P} \neq \mathsf{NP}$.*

PROOF. We present only the proof for $\mathsf{MCSP}^{\mathsf{A}}$; the proof for MKTP^A is identical. Suppose that $\mathsf{P} = \mathsf{NP}$ and $\mathsf{MCSP}^{\mathsf{A}}$ is hard for P under AC^0 reductions. Thus, there is a family $\{C_n\}$ of AC^0 circuits reducing SAT to $\mathsf{MCSP}^{\mathsf{A}}$, such that $C_n(\phi) = f(\phi)$, where f is the reduction function and ϕ is an instance of SAT.

Now we claim that succ.SAT$\leq_m^p$succ.MCSP^A. To see this, consider an instance D of succ.SAT (that is, a circuit D on n variables that, when given input i, outputs the ith bit of a SAT instance of size 2^n). This problem has been shown to be complete for NEXP(Papadimitriou 2003). By Proposition 2.10, we have that the language

$$\{(D,i) \mid \text{the } i\text{th bit of } f(tt(D)) \text{ is } 1\}$$

is in PH. By our assumption that $\mathsf{P} = \mathsf{NP}$, we have that this language is in P. Let $\{E_m\}$ be a family of circuits deciding this language. The function that takes input D and outputs $E_{|(D,n)|}$ (with D hardwired in) is a polynomial-time reduction from succ.SAT to succ.MCSP^A, which is in $(\Sigma_2^p)^A$, by Corollary 2.8. Since $A \in \mathsf{P}$ (by our assumption that $\mathsf{P} = \mathsf{NP}$), we have that $\mathsf{NEXP} \subseteq \mathsf{P}$, which is a contradiction. □

COROLLARY 3.2. *If there is any set $A \in \mathsf{CH}$ such that $\mathsf{MCSP}^{\mathsf{A}}$ (or MKTP^A) is hard for P under TC^0 reductions, then $\mathsf{P} \neq \mathsf{PP}$.*

PROOF. The proof is similar to that of the preceding theorem. If $\mathsf{P} = \mathsf{PP}$, and there is a TC^0 reduction f from SAT to $\mathsf{MCSP}^{\mathsf{A}}$, then the language

$$\{(D, i) \mid \text{the } i\text{th bit of } f(tt(D)) \text{ is } 1\}$$

is in CH (by Proposition 2.11), and hence is in P.

Now, just as above, we use the circuit family recognizing this language to construct a polynomial-time reduction from succ.SAT to succ.MCSP^A, leading to the contradiction that $\mathsf{NEXP} = \mathsf{P}$. □

COROLLARY 3.3. *Suppose that* MCSP *(or* MKTP*) is hard for* P *under logspace many-one reductions. Then* $\mathsf{P} \neq \mathsf{PSPACE}$.

PROOF. The proof proceeds along similar lines. Assume $\mathsf{P} = \mathsf{PSPACE}$. Consider an instance D of succ.SAT, where there is a reduction f computable in logspace reducing SAT to MCSP. Then the language

$$\{(D, i) \mid \text{the } i\text{th bit of } f(tt(D)) \text{ is } 1\}$$

is in PSPACE, since polynomial space suffices in order to compute f on an exponentially-large input. (We don't need to store the string $tt(D)$, the bits of $tt(D)$ can re-computed when they are needed.) By our assumption that $\mathsf{P} = \mathsf{PSPACE}$, this language is in P, and hence is recognized by a uniform circuit family $\{E_m\}$.

Now, as above, the function that maps D to $E_{|(D,n)|}$ (with D hardwired in) is a polynomial-time reduction from succ.SAT to succ.MCSP, which yields the contradiction that $\mathsf{NEXP} = \mathsf{P}$. □

THEOREM 3.4. *Suppose* $\mathsf{MCSP}^{\mathsf{EXP}}$ *is* NP*-hard under polynomial-time reductions. Then* $\mathsf{NEXP} = \mathsf{EXP}$.

PROOF. Let f be the reduction taking an instance of SAT to an instance of $\mathsf{MCSP}^{\mathsf{EXP}}$. We construct a reduction from succ.SAT to B-succ.$\mathsf{MCSP}^{\mathsf{EXP}}$ for some $B \in \mathsf{EXP}$.

Consider the language

$$L = \{(C, i) \mid \text{the } i\text{th bit of } f(\phi_C) \text{ is } 1\},$$

where ϕ_C is the formula described by the circuit C, viewed as an instance of succ.SAT with n input variables. We can decide L

in exponential time because we can write down ϕ_C in exponential time, and then we can compute $f(\phi_C)$ in exponential time because f is a poly-time reduction on an exponentially large instance. Let $\{D_m\}$ be a family of oracle circuits for L, using an oracle for an EXP-complete language B. Thus the mapping $C \mapsto D_{|C|+n}$ is a polynomial-time reduction from succ.SAT to B-succ.$\mathsf{MCSP}^{\mathsf{EXP}}$, which is in $(\Sigma_2^p)^{\mathsf{EXP}} = \mathsf{EXP}$ (see, e.g., (Allender *et al.* 2010, Theorem 24)), and thus $\mathsf{EXP} = \mathsf{NEXP}$. □

COROLLARY 3.5. *For Levin's time-bounded Kolmogorov complexity measure Kt (Levin 1984), suppose that* $\{(x, i) \mid Kt(x) \leq i\}$ *is* NP*-hard under polynomial-time reductions. Then* $\mathsf{NEXP} = \mathsf{EXP}$.

PROOF. As discussed in (Allender *et al.* 2006), there is essentially no difference between $\mathrm{Kt}(x)$ and $\mathrm{KT}^{\mathsf{EXP}}(x)$. Thus the proof is immediate, given the proof of Theorem 3.4. □

THEOREM 3.6. *If* $\mathsf{MCSP}^{\mathsf{QBF}}$ *or* $\mathsf{MKTP}^{\mathsf{QBF}}$ *is hard for* NP *under logspace reductions, then* $\mathsf{NEXP} = \mathsf{PSPACE}$.

PROOF. Let f be the reduction taking an instance of SAT to an instance of $\mathsf{MCSP}^{\mathsf{QBF}}$. We construct a reduction from succ.SAT to QBF-succ.$\mathsf{MCSP}^{\mathsf{QBF}}$.

Consider the language

$$L = \{(C, i) \mid \text{the } i\text{th bit of } f(\phi_C) \text{ is } 1\},$$

where ϕ_C is the formula described by the circuit C, viewed as an instance of succ.SAT with n input variables. We can decide L in PSPACE, because we can compute $f(\phi_C)$ by building the bits of ϕ_C as they are needed. Let $\{D_m\}$ be a family of oracle circuits for L, using an oracle for QBF. Thus the mapping $C \mapsto D_{|C|+n}$ is a polynomial-time reduction from succ.SAT to QBF-succ.$\mathsf{MCSP}^{\mathsf{QBF}}$, which is in $(\Sigma_2^p)^{\mathsf{QBF}} = \mathsf{PSPACE}$, implying $\mathsf{NEXP} = \mathsf{PSPACE}$. □

COROLLARY 3.7. *If* $\mathsf{MCSP}^{\mathsf{QBF}}$ *(or* $\mathsf{MKTP}^{\mathsf{QBF}}$*) is hard for* P *under logspace reductions, then* $\mathsf{EXP} = \mathsf{PSPACE}$.

PROOF. The proof is identical to the proof of the preceding theorem, with NP replaced by P, and with NEXP replaced by EXP, resulting in a reduction from succ.CVP to QBF-succ.$\mathsf{MCSP}^{\mathsf{QBF}}$. □

If we carry out a similar argument, replacing NP with PSPACE, we obtain the contradiction $\mathsf{EXPSPACE} = \mathsf{PSPACE}$, yielding the following.

COROLLARY 3.8. *None of* $\mathsf{MCSP}^{\mathsf{QBF}}$ *and* $\mathsf{MKTP}^{\mathsf{QBF}}$ *is* PSPACE*-hard under logspace reductions.*

3.2. Impossibility of uniform AC^0 reductions.

THEOREM 3.9. *For any language A that is hard for* PH *under* $\mathsf{P/poly}$ *reductions,* $\mathsf{MCSP}^{\mathsf{A}}$ *is* not *hard for* TC^0 *under uniform* AC^0 *reductions.*

The theorem will follow from the next lemma. Recall that LTH (linear-time hierarchy) stands for the linear-time version of the polynomial-time hierarchy PH.

LEMMA 3.10. *Suppose that, for some language A,* $\mathsf{MCSP}^{\mathsf{A}}$ *is* TC^0*-hard under uniform* AC^0 *reductions. Then* $\mathsf{LTH} \not\subseteq \mathsf{io{-}SIZE}^A[2^{\Omega(n)}]$.

PROOF. It is shown in (Agrawal 2011, Theorems 5.1 and 6.2) that if a set is hard for any class $\mathcal{C}$ that is closed under TC^0 reductions under uniform AC^0 reductions, then it is hard under length-increasing (uniform AC^0)-uniform NC^0 reductions. (Although Theorems 5.1 and 6.2 in (Agrawal 2011) are stated only for sets that are *complete* for $\mathcal{C}$, they do hold also assuming only hardness (Agrawal 2014), using exactly the same proofs.) Here, a (uniform AC^0)-uniform NC^0 reduction is a family $\{F_n\}_{n\geq 0}$ of functions such that each F_n is an NC^0 circuit with the property that the direct connection language $DCL = \{(n,t,i,j) |$ gate i of F_n has type t and has an edge leading from gate $j\}$ with n in *unary* is in Dlogtime-uniform AC^0.

Hence, if $\mathsf{MCSP}^{\mathsf{A}}$ is hard for TC^0 under uniform AC^0 reductions, then we get that PARITY is reducible to $\mathsf{MCSP}^{\mathsf{A}}$ under a length-increasing (uniform AC^0)-uniform NC^0 reduction. Such a reduction R maps PARITY instances $x \in \{0,1\}^n$ to $\mathsf{MCSP}^{\mathsf{A}}$ instances (f, s), where f is the truth table of a Boolean function, $f \in \{0,1\}^m$, for some m such that $n \leq m \leq n^{O(1)}$, and s is the size parameter in binary. Since every Boolean function with an m-bit truth table is computable by a Boolean circuit of size at most m, we may assume that $0 \leq s \leq m$, and hence $|s| \leq O(\log n)$.

Being the output of an NC^0 reduction, the binary string s depends on at most $O(\log n)$ bits in the input string x. Imagine fixing these bits in x to achieve the minimum value of the parameter s. Denote this minimum value of s by v. (We do not need v to be efficiently computable in any sense.) We get a *nonuniform* NC^0 reduction from PARITY on $n - O(\log n) \geq n/2$ bit strings to $\mathsf{MCSP}^{\mathsf{A}}$ with the size parameter fixed to the value v.

CLAIM 3.11. *For any language A and any $0 \leq v \leq m$, $\mathsf{MCSP}^{\mathsf{A}}$ on inputs $f \in \{0,1\}^m$, with the size parameter fixed to v, is solved by a DNF formula of size $m \cdot 2^{O(v^2 \log v)}$.*

PROOF (Claim 3.11). Each A-oracle circuit of size v on $\log m$ inputs can be described by a binary string of length at most $O(v^2 \log v)$, since each of v gates has at most v inputs. Thus, there are at most $2^{O(v^2 \log v)}$ Boolean functions on $\log m$ inputs that are computable by A-oracle circuits of size at most v. Checking if any one of these truth tables equals to the input truth table f can be done by a DNF, where we take an OR over all easy functions, and for each easy function we use an AND gate to check equality to the input f. □

We conclude that PARITY on $n/2$-bit strings is solvable by AC^0 circuits of depth 3 and size $m \cdot 2^{O(v^2 \log v)}$. Indeed, each bit of the truth table f is computable by an NC^0 circuit, and hence by a DNF (and a CNF) of constant size. Plugging in these DNFs (or CNFs) for the bits of f into the DNF formula from Claim 3.11 yields the required depth-3 AC^0 circuit for PARITY on inputs of length at least $n/2$.

Next, since PARITY on m-bit strings requires depth-3 AC^0 circuits of size at least $2^{\Omega(\sqrt{m})}$ (Håstad 1989), we get that $v \geq n^{1/5}$. Hence, on input 0^n, our *uniform* NC^0 reduction produces (f, s) where f is the truth table of a Boolean function on r-bit inputs that has A-oracle circuit complexity at least $v \geq n^{1/5} \geq 2^{\epsilon r}$, for some $\epsilon > 0$.

Finally, since the NC^0 reduction is (uniform AC^0)-uniform, we get that the Boolean function whose truth table is f is computable in LTH. □

Proof (Theorem 3.9). Towards a contradiction, suppose that MCSP^A is TC^0-hard under uniform AC^0 reductions. Then, by Lemma 3.10, there is a language $L \in \mathsf{PH}$ that requires A-oracle circuit complexity $2^{\Omega(n)}$ almost everywhere. However, since A is PH-hard under $\mathsf{P/poly}$ reductions, we get that $L \in \mathsf{SIZE}^A[\mathsf{poly}]$. A contradiction. □

Corollary 3.12. $\mathsf{MCSP}^{\oplus\mathsf{P}}$ *is not* TC^0*-hard under uniform* AC^0 *reductions.*

Proof. By Toda's theorem (Toda 1991), $\mathsf{PH} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}}$, which in turn is contained in the class of problems $\mathsf{P/poly}$-reducible to the standard complete problem for $\oplus\mathsf{P}$. The result then follows by Theorem 3.9. □

Corollary 3.13. *Suppose that, for some oracle* A*,* MCSP^A *is* TC^0*-hard under uniform* AC^0 *reductions. Then* $\mathsf{NP}^A \not\subseteq \mathsf{SIZE}^A[\mathsf{poly}]$.

Proof. If $\mathsf{NP}^A \subseteq \mathsf{SIZE}^A[\mathsf{poly}]$, then $\mathsf{PH}^A \subseteq \mathsf{SIZE}^A[\mathsf{poly}]$. Now the result follows from Lemma 3.10. □

Remark 3.14. *Murray & Williams (2015) prove results similar to (and implied by) our Lemma 3.10 and Corollary 3.13 for the case of the empty oracle* $A = \emptyset$*. Namely, they show that if* MCSP *is* NP*-hard under uniform* AC^0 *reductions, then* $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ *and* $\mathsf{E} \not\subseteq \mathsf{io{-}SIZE}[2^{\Omega(n)}]$.

Finally, we note that the ideas in our proof of Lemma 3.10 yield an alternate proof of the result by Murray & Williams (2015) that PARITY is not reducible to MCSP via "local" $O(n^{1/2-\epsilon})$-time reductions. We prove the version for polylogtime-uniform NC^0 reductions, but the same argument applies also to the "local" reductions of (Murray & Williams 2015).

THEOREM 3.15 (Murray & Williams 2015). *No polylogtime-uniform* NC^0 *reduction exists from* PARITY *to* MCSP.

PROOF. Suppose there is such a reduction. Similarly to the proof of Lemma 3.10, we conclude that this NC^0 reduction maps 0^n to an MCSP instance (f, s) where f is the truth table of a Boolean function on $r := O(\log n)$ inputs that requires exponential circuit size $s \geq 2^{\Omega(r)}$. On the other hand, since our NC^0 reduction is polylogtime-uniform, the Boolean function with the truth table f is computable in P, and hence in SIZE[poly]. A contradiction. □

3.3. Gap MCSP. For $0 < \epsilon < 1$, we consider the following *gap version* of MCSP, denoted ϵ-gap MCSP: Given (f, s), output 'No' if f requires circuits of size at least s, and output 'Yes' if f can be computed by a circuit of size at most $(1 - \epsilon)s$.

For $\alpha : \mathbb{N} \to \mathbb{R}^+$, call a mapping $R : \{0,1\}^n \to \{0,1\}^m$ α-*stretching* if $m \leq \alpha(n) \cdot n$. We will prove that there is no n^δ-stretching nonuniform AC^0 reduction from PARITY to ϵ-gap MCSP, for certain parameters $0 < \epsilon, \delta < 1$. First, we rule out nonuniform NC^0 reductions.

THEOREM 3.16. *For every* $n^{-1/6} < \epsilon < 1$ *and for every constant* $\delta < 1/30$, *there is* no n^δ*-stretching (nonuniform)* NC^0 *reduction from* PARITY *to* ϵ*-gap* MCSP.

PROOF. Towards contradiction, suppose there is an n^δ-stretching NC^0 reduction from PARITY on inputs $x \in \{0,1\}^n$ to ϵ-gap MCSP instances (f, s). Fix to zeros all $O(\log n)$ bit positions in the string x that determine the value of the size parameter s. As in the proof of Lemma 3.10, we get an NC^0 reduction from PARITY on at least $n/2$ bits y to the ϵ-gap MCSP instance with the size parameter fixed to some value at least $v \geq n^{1/5}$. (Recall that $v \geq n^{1/5}$ is the *minimum* value of the size parameter s on any input x; so, in particular, we get at least this value v on the string of all zeros in the positions that determine the value of s.)

By our assumption, $|f| \leq n \cdot n^{\delta}$. Since each bit of f is computable by an NC^0 circuit, we get that each bit of f depends on at most c bits in the input y. The total number of pairs (i, j) where f_i depends on bit y_j is at most $c \cdot |f|$. By averaging, there is a bit y_j, $1 \leq j \leq n/2$, that influences at most $c|f|/(n/2) \leq 2cn^{\delta}$ bit positions in the string f.

Fix y so that all bits are 0 except for y_j (which is set to 1). This y is mapped by our NC^0 reduction to the truth table f' that is computable by a circuit of size at most $(1-\epsilon)v$. On the other hand, flipping the bit y_j to 0 forces the reduction to output a truth table f'' of circuit complexity at least v. But, y_j influences at most $2cn^{\delta}$ positions in f', and so the circuit complexity of f'' differs from that of f' by at most $O(n^{\delta} \log n)$ gates (as we can just construct a "difference" circuit of that size that is 1 on the at most $2cn^{\delta}$ affected positions of f'). We get $\epsilon v \leq O(n^{\delta} \log n)$, which is impossible when $\delta < 1/30$. □

Now we extend Theorem 3.16 to the case of nonuniform AC^0 reductions.

THEOREM 3.17. *For every $n^{-1/7} < \epsilon < 1$ and for every constant $\delta < 1/31$, there is* no *n^{δ}-stretching (nonuniform) AC^0 reduction from* PARITY *to ϵ-gap* MCSP.

PROOF. Towards contradiction, suppose there is a n^{δ}-stretching AC^0 reduction from PARITY on n-bit strings to the ϵ-gap MCSP. We will show that this implies the existence of an NC^0 reduction with parameters that contradict Theorem 3.16 above.

CLAIM 3.18. *For every constant $\gamma > 0$, there exist a constant $a > 0$ and a restriction of our AC^0 circuit satisfying the following:*

(i) each output of the restricted circuit depends on at most a inputs, and

(ii) the number of unrestricted variables is at least $n^{1-\gamma}$.

PROOF (Claim 3.18). Recall that a random p-restriction of n variables $x_1, \ldots, x_n$ is defined as follows: for each $1 \leq i \leq n$, with probability p, leave x_i unrestricted, and with probability $1 - p$, set x_i to 0 or 1 uniformly at random. By Håstad's Switching Lemma (Håstad 1989), the probability that a given CNF on n variables with bottom fan-in at most t does not become a decision tree of depth at most r after being hit with a random p-restriction is at most $(5pt)^r$.

For an AC^0 circuit of size n^k and depth d, set

$$p := (5a)^{-1} \cdot n^{-2k/a}$$

for some constant $a > 0$ to be determined. Applying this random p-restriction d times will reduce the original circuit to a decision tree of depth a with probability at least

$$1 - dn^k(5pa)^a > 3/4.$$

The expected number of unrestricted variables at the end of this process is

$$\begin{aligned} p^d n &\geq \Omega(n/n^{2kd/a}) \\ &= \Omega(n/n^{\gamma'}), \end{aligned}$$

for $\gamma' := 2kd/a$. By Chernoff bounds, the actual number of unrestricted variables is at least 1/2 of the expectation with probability at least 3/4.

Thus, with probability at least 1/2, we get a restriction that makes the original AC^0 circuit into an NC^0 circuit on at least $n/n^{2\gamma'}$ variables, where each output of the new circuit depends on at most a input variables. Setting $\gamma := 2\gamma'$, we get that $a = (4kd)/\gamma$. □

We get an NC^0 reduction from PARITY on $n' := n^{1-\gamma}$ variables to ϵ-gap MCSP. This reduction is at most $(n')^{(\delta+\gamma)/(1-\gamma)}$-stretching. But, for any $0 < \gamma < (1/31)^2$ so that $(\delta+\gamma)/(1-\gamma) < 1/30$ and $\epsilon > n^{-1/7} > (n')^{-1/6}$, such a reduction cannot exist by Theorem 3.16. □

4. Generalizations

Theorem 3.1 gives consequences of MCSP being hard for P. The property of P that is exploited in the proof is that the polynomial hierarchy collapses to P if $\mathsf{NP} = \mathsf{P}$. (This is required so that we can efficiently obtain a circuit that computes bits of the reduction, knowing only that it is in the polynomial hierarchy.)

The next theorem formalizes this observation:

THEOREM 4.1. *Let $\mathcal{C}$ be any class such that if $\mathsf{NP} = \mathcal{C}$, then $\mathsf{PH} = \mathcal{C}$. If there is a set $A \in \mathsf{PH}$ that is hard for $\mathcal{C}$ under $\leq_T^p$ reductions such that $\mathsf{MCSP}^{\mathsf{A}}$ (or MKTP^A) is hard for $\mathcal{C}$ under uniform AC^0 reductions, then $\mathsf{NP} \neq \mathcal{C}$.*

PROOF. Suppose that $\mathsf{NP} = \mathcal{C}$, and $\mathsf{MCSP}^{\mathsf{A}}$ is hard for $\mathcal{C}$. Then, there exists a reduction from SAT to $\mathsf{MCSP}^{\mathsf{A}}$ computable in AC^0. As in the proof of Theorem 3.1, we can use this to construct a $\leq_T^p$ reduction from succ.SAT to B-succ.MCSP^A for some B in PH; and thus B is in $\mathcal{C}$ by our assumption. Thus $B \leq_T^p A$. By Corollary 2.8 this implies that succ.SAT is in $(\Sigma_2^p)^A$, which is in the polynomial hierarchy, and hence is in NP.

However, this implies $\mathsf{NEXP} \subseteq \mathsf{NP}$, which contradicts the Nondeterministic Time Hierarchy Theorem (Seiferas *et al.* 1978). □

COROLLARY 4.2. *Let A be any set in the polynomial hierarchy. If $\mathsf{MCSP}^{\mathsf{A}}$ (or MKTP^A) is hard for $\mathsf{AC}^0[6]$ under AC^0 reductions, then $\mathsf{AC}^0[6] \neq \mathsf{NP}$.*

Recall that SZK denotes the class of languages with Statistical Zero-Knowledge proofs.

COROLLARY 4.3. *Let A be any set in the polynomial hierarchy that is hard for SZK under $\leq_T^p$ reductions. If $\mathsf{MCSP}^{\mathsf{A}}$ is hard for SZK under AC^0 reductions, then $\mathsf{SZK} \neq \mathsf{NP}$.*

PROOF. SZK is closed under complementation (Goldreich *et al.* 1998; Okamoto 2000). Thus if NP is equal to the class of languages in SZK, then $\mathsf{coNP} = \mathsf{NP} = \mathsf{SZK}$ and PH collapses to SZK. Thus SZK satisfies the hypothesis of Theorem 4.1. □

Similarly, we can state the following theorem about TC^0 reductions.

Theorem 4.4. *Let $\mathcal{C}$ be any class such that if* $\mathsf{PP} = \mathcal{C}$, *then* $\mathsf{CH} = \mathcal{C}$. *If there is a set* $A \in \mathsf{CH}$ *that is hard for* $\mathcal{C}$ *under* $\leq_T^p$ *reductions such that* $\mathsf{MCSP}^{\mathsf{A}}$ *(or* MKTP^A*) is hard for* $\mathcal{C}$ *under uniform* TC^0 *reductions, then* $\mathsf{PP} \neq C$.

Proof. Suppose that $\mathsf{PP} = \mathcal{C}$, and that $\mathsf{MCSP}^{\mathsf{A}}$ is hard for $\mathcal{C}$. Then, there exists a reduction from Maj.SAT (the standard complete problem for PP) to $\mathsf{MCSP}^{\mathsf{A}}$ computable in TC^0. Similarly to Corollary 3.2, this gives us a $\leq_T^p$ reduction from succ.MajSAT to B-succ.MCSP^A for some $B \in \mathsf{CH}$; and thus B is in $\mathcal{C}$. Then, $B \leq_T^p A$, and thus succ.MajSAT is in $(\Sigma_2^p)^A$, which is in CH, and hence is in PP. However, succ.MajSAT is complete for probabilistic exponential time, and hence is not in PP. □

Fenner *et al.* (1994) introduced several complexity classes, including SPP and WPP that are "low for PP", in the sense that $\mathsf{PP} = \mathsf{PP}^{\mathsf{SPP}} = \mathsf{PP}^{\mathsf{WPP}}$. Thus we obtain the following corollary:

Corollary 4.5. *Let A be any set in the counting hierarchy that is hard for* WPP *under* $\leq_T^p$ *reductions. If* $\mathsf{MCSP}^{\mathsf{A}}$ *is hard for* WPP *(or* SPP*) under uniform* TC^0 *reductions, then* $\mathsf{WPP} \neq \mathsf{PP}$ *(respectively* $\mathsf{SPP} \neq \mathsf{PP}$*).*

5. Discussion

The contrast between Theorem 3.1 and Corollary 3.7 is stark. Theorem 3.1 obtains a very unsurprising consequence from the assumption that MCSP is hard for P under a very restrictive class of reductions, while Corollary 3.7 obtains a very unlikely collapse from the assumption that the apparently much harder problem $\mathsf{MCSP}^{\mathsf{QBF}}$ is hard for P under a much less restrictive class of reductions. Yet, the absence of any known efficient reduction from MCSP to $\mathsf{MCSP}^{\mathsf{QBF}}$ means that we have been unable to obtain any *unlikely* consequences by assuming that MCSP is hard for P. We believe that it should be possible to provide evidence that MCSP is not hard for P, and we pose this as an open question for further research.

Acknowledgements

This research was supported in part by NSF grants CCF-1064785, CCF-1423544, and CCF-1555409, and by an NSERC Discovery Grant. Some of this work was carried out at the 2014 Dagstuhl Workshop on Algebra in Computational Complexity (Dagstuhl Seminar 14391), and was performed while the second author was an undergraduate student at the California Institute of Technology. An extended abstract of this paper has appeared as (Allender *et al.* 2015). We acknowledge helpful discussions with Ryan Williams, Chris Umans, Manindra Agrawal, and Mitsunori Ogihara. We also thank the anonymous referees for their helpful remarks and suggestions.

References

Manindra Agrawal (2011). The isomorphism conjecture for constant depth reductions. *Journal of Computer and System Sciences* **77**(1), 3–13.

Manindra Agrawal (2014). Personal Communication.

Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek & Detlef Ronneburger (2006). Power from random strings. *SIAM Journal on Computing* **35**(6), 1467–1493.

Eric Allender & Bireswar Das (2014). Zero Knowledge and Circuit Minimization. In *Mathematical Foundations of Computer Science (MFCS)*, volume 8635 of *Lecture Notes in Computer Science*, 25–32. Springer.

Eric Allender & Vivek Gore (1991). Rudimentary Reductions Revisited. *Information Processing Letters* **40**(2), 89–95.

Eric Allender & Vivek Gore (1993). On strong separations from AC^0. In *Advances in Computational Complexity Theory*, Jin-Yi Cai, editor, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 21–37. AMS Press.

Eric Allender, Dhiraj Holden & Valentine Kabanets (2015). The Minimum Oracle Circuit Size Problem. In *32nd International*

Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany, Ernst W. Mayr & Nicolas Ollinger, editors, volume 30 of *LIPIcs*, 21–33. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-78-1. URL http://dx.doi.org/10.4230/LIPIcs.STACS.2015.21.

Eric Allender, Michal Koucký, Detlef Ronneburger & Sambuddha Roy (2010). The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences* **77**, 14–40.

José L Balcázar, Antoni Lozano & Jacobo Torán (1992). The complexity of algorithmic problems on succinct instances. In *Computer Science*, 351–377. Springer.

Stephen A. Fenner, Lance Fortnow & Stuart A. Kurtz (1994). Gap-Definable Counting Classes. *Journal of Computer and System Sciences* **48**(1), 116–148.

Lance Fortnow (2000). Time-Space Tradeoffs for Satisfiability. *Journal of Computer and System Sciences* **60**(2), 337–353.

Hana Galperin & Avi Wigderson (1983). Succinct representations of graphs. *Information and Control* **56**(3), 183–198.

Oded Goldreich, Amit Sahai & Salil Vadhan (1998). Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 399–408. ACM.

Johan Håstad (1989). Almost optimal lower bounds for small depth circuits. In *Randomness and Computation*, S. Micali, editor, 143–170. Advances in Computing Research, vol. 5, JAI Press, Greenwich, Connecticut.

Valentine Kabanets & Jin-Yi Cai (2000). Circuit minimization problem. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 73–79. ACM.

Leonid Levin (1984). Randomness Conservation Inequalities; Information and Independence in Mathematical Theories. *Information and Control* **61**, 15–37.

PIERRE MCKENZIE, MICHAEL THOMAS & HERIBERT VOLLMER (2010). Extensional uniformity for boolean circuits. *SIAM Journal on Computing* **39**(7), 3186–3206.

CODY MURRAY & RYAN WILLIAMS (2015). On the (Non) NP-Hardness of Computing Circuit Complexity. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, 365–380. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. URL `http://dx.doi.org/10.4230/LIPIcs.CCC.2015.365`.

TATSUAKI OKAMOTO (2000). On Relationships between Statistical Zero-Knowledge Proofs. *Journal of Computer and System Sciences* **60**(1), 47–108.

CHRISTOS H. PAPADIMITRIOU (2003). *Computational complexity*. John Wiley and Sons Ltd.

CHRISTOS H. PAPADIMITRIOU & MIHALIS YANNAKAKIS (1986). A note on succinct representations of graphs. *Information and Control* **71**(3), 181–185.

JOEL I. SEIFERAS, MICHAEL J. FISCHER & ALBERT R. MEYER (1978). Separating Nondeterministic Time Complexity Classes. *Journal of the ACM* **25**(1), 146–167.

RAYMOND M. SMULLYAN (1961). Theory of Formal systems. In *Annals of Math. Studies* 47. Princeton University Press.

SEINOSUKE TODA (1991). PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing* **20**(5), 865–877.

JACOBO TORÁN (1991). Complexity classes defined by counting quantifiers. *Journal of the ACM* **38**(3), 752–773.

BORIS A. TRAKHTENBROT (1984). A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Annals of the History of Computing* **6**(4), 384–400.

KLAUS W WAGNER (1986). The complexity of combinatorial problems with succinct input representation. *Acta Informatica* **23**(3), 325–356.

Ryan Williams (2012). URL http://cstheory.stackexchange.com/questions/10320/succinct-problems-in-mathsfp/10546#10546.

Celia Wrathall (1978). Rudimentary predicates and relative computation. *SIAM Journal on Computing* **7**(2), 194–209.

Manuscript received 12 May, 2015

Eric Allender
Department of Computer Science
Rutgers University
Piscataway, NJ, USA
allender@cs.rutgers.edu

Dhiraj Holden
CSAIL
Massachusetts Institute of Technology
Cambridge, MA, USA
dholden@mit.edu

Valentine Kabanets
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
kabanets@cs.sfu.ca