Circuit Lower Bounds from NP-hardness of MCSP under Turing Reductions

Michael Saks
Rutgers University, United States of America
saks@math.rutgers.edu

Rahul Santhanam
University of Oxford, United Kingdom
rahul.santhanam@cs.ox.ac.uk

Abstract

The fundamental Minimum Circuit Size Problem is a well-known example of a problem that is neither known to be in P nor known to be NP-hard. Kabanets and Cai [18] showed that if MCSP is NP-hard under "natural" m-reductions, superpolynomial circuit lower bounds for exponential time would follow. This has triggered a long line of work on understanding the power of reductions to MCSP.

Nothing was known so far about consequences of NP-hardness of MCSP under general Turing reductions. In this work, we consider two structured kinds of Turing reductions: *parametric honest* reductions and *natural* reductions. The latter generalize the natural reductions of Kabanets and Cai to the case of Turing-reductions. We show that NP-hardness of MCSP under these kinds of Turing-reductions imply superpolynomial circuit lower bounds for exponential time.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Circuit complexity; Theory of computation → Complexity classes

Keywords and phrases Minimum Circuit Size Problem, Turing reductions, circuit lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.26

Funding *Rahul Santhanam*: This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement No. 615075.

1 Introduction

The Minimum Circuit Size Problem (MCSP) is a fundamental problem in computational complexity that has been studied in various forms since the 1950s [24, 1]. Given the truth table of a Boolean function f and a parameter s, the question is whether f has Boolean circuits of size at most s. It is easy to see that MCSP is in NP: we can simply guess a circuit C of size at most s, and verify that for each input of f that C computes f correctly. Since we are given the truth table of f explicitly, this verification can be done in polynomial time.

MCSP is a rare example of a natural problem in NP for which we neither know that the problem is in P nor that it is NP-complete. There is some evidence that the problem is not in P. The celebrated results on "natural proofs" by Razborov and Rudich [23] can be interpreted as saying that MCSP is not in P if one-way functions exist [18, 2].

Regarding the question of NP-completeness, however, the evidence is more mixed. It is reported that Levin delayed publishing his seminal results on NP-completeness because he was hoping to show that MCSP is NP-complete [8]. More than 50 years later, we still have no idea.

It has been known for a while that the analogue of MCSP for DNFs, where we ask whether the function corresponding to a given truth table has small DNFs, is NP-complete [20, 10, 5]. Intuitively it would appear that checking if a given function has small circuits is at least as

hard a problem as checking if a given function has small DNFs. But it does not seem easy to make this intuition precise in terms of giving a reduction from the latter to the former[1].

Looking at the proof that the analogue of MCSP for DNFs is NP-hard, one notices that an essential component of the reduction is an explicit function, namely Parity, that is hard for DNFs. This leads one to wonder whether the difficulty of coming up with an NP-hardness reduction for MCSP is related to the difficulty of proving Boolean circuit lower bounds for explicit functions.

Kabanets and Cai show that the answer is yes, in an influential paper [18] which led to a resurgence of interest in MCSP among complexity theorists. They define a notion of "natural reduction" to MCSP. A natural reduction is a many-one reduction in which the parameter as well as the size of the output depend only on the size of the input. Kabanets and Cai observe that standard reductions showing NP-hardness of various well-studied problems are natural in this sense. They then show that any natural reduction of SAT to MCSP implies that exponential time requires super-polynomial size Boolean circuits. Note that this does not give evidence *against* NP-hardness. However, given the well-known difficulty of proving circuit lower bounds for exponential time, it does indicate that *arguing* for NP-hardness of MCSP via natural reductions is likely to be difficult.

The work of Kabanets and Cai has led to a long line of results on reductions to MCSP and their implications[2, 3, 4, 7, 21, 17, 16, 6, 22, 15, 14]. Much of this work has focused on reductions implementable within low-depth complexity classes [7, 21, 6, 22] or on many-one and truth-table reductions [7, 21, 17, 16].

In this work, our focus is on polynomial-time Turing reductions to MCSP, for which very little is known. On the one hand, we are motivated by the question of NP-hardness of MCSP, and whether more powerful reductions help in this regard. Allender and Das [3] showed that every problem in Statistical Zero Knowledge reduces to MCSP via probabilistic Turing reductions, so at least for some problems that are believed to be hard, Turing reductions have been found to be useful in reducing to MCSP.

On the other hand, from the perspective of complexity theorists who care about circuit lower bounds, we find the connections between reductions to MCSP and circuit lower bounds intriguing, and would like to understand the extent to which these connections hold.

1.1 Our Results

We consider two kinds of structured Turing reductions - *parametric honest* reductions, which were defined in the context of many-one reductions by Hitchcock and Pavan [17], and *natural* reductions, which generalize the notion defined by Kabanets and Cai [18]. Parametric honest reductions are Turing reductions where there is a polynomial lower bound on the parameter of each query, as a function of input size. Natural reductions are Turing reductions where the parameter of any query made on an input only depends on the size of the input.

Our first main result shows that NP-completeness of MCSP under parametric honest Turing reductions implies super-polynomial circuit lower bounds for E.

▶ **Theorem 1.** *If* MCSP *is* NP*-hard under parametric honest Turing reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.

We remark that every language in P reduces to MCSP under parametric honest Turing reductions, so we crucially need to exploit the assumption that we reduce from hard languages

[1] Similar issues come up when trying to prove hardness of properly learning Boolean circuits.

in NP in order to derive the conclusion of Theorem 1.

Our second main result shows that NP-completeness of MCSP under natural Turing reductions implies super-polynomial circuit lower bounds for E.

▶ **Theorem 2.** *If* MCSP *is* NP*-hard under natural Turing reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.

The proof of Theorem 2 builds upon the ideas of Theorem 1, but is significantly more involved technically, and requires several new ideas.

We next describe the main ideas behind our proofs.

1.2 Main Ideas

Our starting point is the idea of Kabanets and Cai [18] for deriving circuit lower bounds from structured many-one reductions. Suppose we could efficiently generate NO instances of any length for some language L that reduces to MCSP via an m-reduction. We could hope to efficiently generate truth tables of hard functions as follows: we generate a NO instance x_n of L of length n, and then apply the m-reduction f from L to MCSP to generate a truth table y_n and a parameter s_n. Since x_n is a NO instance of L and f is an m-reduction, we are guaranteed that y does not have circuits of size s.

If s is large, say at least n^ϵ for some constant ϵ, then we are done, since we have efficiently generated a truth table y of a Boolean function that does not have sub-exponential size Boolean circuits as a function of its input length. But a priori we do not have control over the parameter s associated with y. Imagine for example a reduction that magically knows that each x_n is a NO instance and therefore produces s_n so small that it gives us no non-trivial information about circuit lower bounds.

Kabanets and Cai get around this problem by considering natural reductions, where the parameter s_n depends only on the input length n. Now there are two cases: either all but finitely many input lengths yield small parameter $s_n \leq \log(n)^{\log\log(n)}$, or infinitely many input lengths yield parameter $s_n > \log(n)^{\log\log(n)}$. In the first case, Kabanets and Cai use a standard indirect diagonalization argument to argue that circuit lower bounds follow. In the second case, we directly get hard truth tables for infinitely many n by composing the reduction with the efficient generation of NO instances. Thus, in either case, we get $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.

Note that in the case where infinitely many input lengths yield hard parameters, the Kabanets-Cai argument does not actually exploit the hardness of the language L from which we are reducing. The argument works equally well starting from a language in polynomial time. When we deal with Turing reductions rather than m-reductions, this feature of their argument is an issue, since any language L in P can be decided trivially by Turing reductions that ignore the answers to any queries they ask and simulate the polynomial-time algorithm for L directly to arrive at an answer.

Let us consider first the case of parametric honest reductions. Parametric honesty means that we don't need to worry about the parameter being small. Suppose that SAT reduces to MCSP via a parametric honest Turing reduction implemented by an oracle machine M. Our first idea is simple: we define a simulation A of SAT which simply runs M and answers all queries of M by 'yes'. This algorithm A clearly runs in polynomial time. If it is correct on all but finitely many instances, we have that $\mathsf{NP} = \mathsf{P}$ and we can then derive circuit lower bounds using a standard indirect diagonalization technique.

If A is wrong on infinitely many instances, however, it is not obvious how to use this. Here we exploit the main idea of Gutfreund, Shaltiel and Ta-Shma [12], who show how any efficient algorithm A that fails to solve SAT can be used to efficiently produce, for infinitely

many n, a small set of instances S of sizes polynomially related to n, such that A fails on some instance in S. This is very useful for us, as an instance on which A fails must ask a query to M that is answered incorrectly. But since we always answer queries by 'yes' in A, a query is only answered incorrectly if the true answer is "no", i.e., if the string y that is queried is the truth table of a hard Boolean function. We don't know which of the queries this is, but we do know that there is a *first* wrongly answered query, and that by the parametric honesty condition, this query has large parameter. Hence, by simply concatenating the queries asked by A on the instances in S, we obtain hard truth tables of size poly(n) for infinitely many n, and this enables us to conclude that E requires super-polynomial size circuits.

This is the argument for the proof of Theorem 1. The assumption of parametric honesty is an important part of this argument. For the proof of Theorem 2, where we consider natural reductions, we need to work much harder and use several new ideas.

We would like to adopt the following strategy. Suppose there is a natural Turing-reduction from SAT to MCSP. Choose a "threshold" parameter s and try to solve SAT by running the reduction and answering all queries with parameter less than the threshold correctly by doing brute force search, and all queries with parameter greater than the threshold by 'yes'.

Either this simulation succeeds, or it fails. If it succeeds, and the threshold parameter s is small, NP is easy, and using the standard indirect diagonalization argument, we get that E requires large circuits. If it fails, we would like to argue that we can efficiently find instances on which the simulation fails. Such an instance must ask queries with parameters larger than the threshold that are wrongly answered 'yes', so the concatenation of such queries must be a hard truth table.

To efficiently find instances on which the simulation fails, we can again to try to use the strategy of [12]. Unfortunately, there is a catch. The time taken by the algorithm of [12] to find hard instances is at least the time taken for the simulation, which is exponential in s. It might be that the parameter for queries on these instances is just slightly larger than s. Thus we will take time exponential in s to find truth tables requiring circuits only slightly larger than s, which won't give us sufficiently strong circuit lower bounds for E.

To remedy this issue, we consider two different simulations, M and M'. M is guaranteed to be correct and proceeds by just evaluating queries using brute force search. If M is relatively efficient, we are in good shape, as we can use indirect diagonalization to get the consequence we want. If M is not always efficient, we get an infinite sequence of input lengths on which large parameter queries are made. Here "large" means at least s, where s is super-polylogarithmic in n, but still small enough so that the indirect diagonalization helps us get circuit lower bounds in case M succeeds.

In fact, by using the paddability of SAT in our argument, we get not just an infinite sequence of input lengths yielding large parameters, but a sequence of long intervals, within which all input lengths yield large parameters. This will be important later, as the [12] algorithm does not produce hard instances at a given input length, but rather at one of two input lengths which are polynomially related.

The simulation M' will use a smaller parameter threshold s', chosen so that s' is superpolylogarithmic in n, but also so that s is superpolynomial in s'. The simulation M' implements our initial idea: we answer queries with parameter at most s' by brute force search, and other queries by 'yes'.

Now either this simulation is correct on all but finitely many input lengths, or it fails on infinitely many. If it is correct on all but finitely many input lengths, we can use indirect diagonalization as before, but it is not clear how to use the failure on infinitely many, without more structural information on where the failures occur.

So we argue instead as follows, using a notion of "robust simulation" introduced in [11]. If the simulation is correct, robustly often, than we can carry the indirect diagonalization through. Otherwise, the simulation fails on at least one input length in every large enough sequence of input lengths. Using the large stretches of input lengths with large parameters from our first simulation, we get that there are infinitely many long sequences of input lengths where the simulation M' fails and the corresponding parameters of queries are large.

Now we use the strategy of [12]. We use simulation M' itself to find a small set of inputs on which M' fails, and on which the parameter of correctly produced queries is at least s. M' will take time only exponential in s', and since s is super-polynomial in s', the time taken to find this small set of inputs will be sub-exponential in s. By concatenating the queries made on this small set of inputs, we get a hard truth table, and hence can arrive at our desired conclusion.

There are numerous technical details which we have ignored in the above discussion, but hopefully this description helps in understanding the proof of Theorem 2.

1.3 Related Work

As mentioned earlier in the Introduction, there have been several works in recent years studying reductions to MCSP and their consequences. We briefly survey this work in the context of our results. We focus on work that either shows interesting consequences of hardness for MCSP under certain types of reductions, or unconditionally rules out hardness, as this is closest in spirit to our work here.

Kabanets and Cai [18] defined natural m-reductions, and showed that NP-hardness of MCSP under natural m-reductions implies super-polynomial circuit lower bounds for E. Murray and Williams [21] showed unconditionally that MCSP is unconditionally not NP-hard under very efficient local reductions where each output bit only depends on a limited number of input bits. In fact, their technique even rules out P-hardness of MCSP under local reductions. They also show that NP-hardness of MCSP under m-reductions (without the "natural" constraint of [18]) implies EXP $\neq$ ZPP. Hitchcock and Pavan showed that NP-hardness of MCSP under general truth-table reductions implies EXP $\neq$ ZPP. Allender, Holden and Kabanets [7] consider the question of hardness for various oracle versions of MCSP and show some unlikely consequences of hardness under efficient reductions when the oracle is powerful.

The work that is closest to ours is that of Hirahara and Watanabe [16], who also consider Turing-reductions to MCSP. They consider a kind of reduction they call "oracle-independent", and show that no language outside of P reduces to MCSP using an oracle-independent reduction. However, there are examples of useful reductions known that are not oracle-independent [6] [2]. It is also shown in [16] that Turing-hardness of approximating the minimum circuit size implies EXP $\neq$ ZPP. We note that the hardness of approximation factors required for this result are large - the assumption is that it is hard to approximate the logarithm of the circuit size to within any constant factor. Moreover, [16] do not derive a circuit lower bound from this assumption, but the weaker statement that EXP $\neq$ ZPP.

[2] On the other hand, we are not aware of any hardness results for MCSP using reductions that are not natural.

2 Preliminaries

2.1 Basic Notions

We refer to the book by Arora and Barak [9] for definitions of standard complexity classes.

Given a circuit class $\mathfrak{C}$, we use $\mathfrak{C}[s(n)]$ to refer to the class of functions computed by $\mathfrak{C}$-circuits of size $s(n)$.

Given a language $L \subseteq \{0,1\}^*$, co-L denotes the complement of L. Given language L and $n \in \mathbb{N}$, L_n is used to refer to $L \cap \{0,1\}^n$.

Given a set $S \subseteq \mathbb{N}$ and languages L and L', we say L agrees with L' on S if for all $n \in S$, $L_n = L'_n$.

A time-constructible function $T : \mathbb{N} \to \mathbb{N}$ is a function such that there is a Turing machine transducer M, which for each n, on input 1^n halts with output $T(n)$ within $O(T(n))$ steps.

We need a generalization of the notion of robustly-often simulation from [11]. Given $g : \mathbb{N} \to \mathbb{N}$, a set $S \subseteq \mathbb{N}$ is called *g-robust* if for each constant k, there is $m \in \mathbb{N}$ such that for all n such that $m \leq n \leq g(m)^k$, $n \in S$. Note that any g-robust set is also poly(g)-robust.

Given a function g, a language L and a complexity class $\mathcal{C}$, we say L is g-robustly often in $\mathcal{C}$ if there is $L' \in \mathcal{C}$ for which there is a g-robust set S such that L agrees with L' on S. Given function g and complexity classes $\mathcal{C}$ and $\mathcal{C}'$, we say that $\mathcal{C}$ is g-robustly often in $\mathcal{C}'$ if for all languages $L \in \mathcal{C}$, L is g-robustly often in $\mathcal{C}'$. We use the term "robustly often" as a synonym for g-robustly often with g the identity function.

The proposition below is a parameterized version of Theorem 12 in [11].

▶ **Proposition 3.** *Let T be a time-constructible function such that $T(n) \geq n$ for all n. If* SAT *is $T(\text{poly}(n))$-robustly often in* DTIME(T)*, then* Σ_2P *is robustly often.in* DTIME($T(\text{poly}(T(\text{poly}(n))))$).

The following is a simple application of standard diagonalization [13].

▶ **Proposition 4.** *Let T be a time-construtible function such that $T = o(2^n)$. Then* E *is not robustly often in* DTIME(T).

2.2 Resource-Bounded Kolmogorov Complexity

We study various notions of resource-bounded Kolmogorov complexity, and associated decision problems.

Fix a universal Turing machine U. The Kt complexity of a string is defined as follows: $\mathsf{Kt}(x) = min\{|p| + \log(t) | U(p)$ halts and outputs x within t steps$\}$.

In the MKtP problem, the instance is a string x together with a parameter s, and the question is whether $\mathsf{Kt}(x) \leq s$.

It is known that MKtP is complete for E under polynomial-size truth-table reductions [2], but completeness under uniform polynomial-time reductions is unknown.

In the MCSP problem, the instance is a string x together with a parameter s, and the question is whether x is the truth table of a Boolean function with circuit complexity at most s.

MCSP is easily seen to be in NP, but it is unknown whether MCSP is NP-hard. A brute-force search strategy of trying all circuits C of size at most s and checking if any of them computes the function with truth table x can easily be implemented to run in time poly$(|x|)s^{O(s)}$.

2.3 Reductions

We will work with constrained notions of polynomial-time Turing reduction.

▶ **Definition 5.** *A polynomial-time Turing reduction from a language L to* MCSP *or* MKtP *is called parametric honest if there is a constant $\epsilon > 0$ such that for every $x \in \{0,1\}^*$, every query made by the reduction on x has parameter bounded below by $|x|^\epsilon$.*

Note that parametric honesty is not implied by the standard notion of honesty for polynomial-time reductions, which only requires the output length to be bounded below by some polynomial in the input length. Implicit in the definition is that the lower bound on parameters of queries is only required to hold when previous queries are answered correctly; when queries are answered wrongly, "all bets are off".

▶ **Definition 6.** *A polynomial-time Turing reduction from a language L to* MCSP *or* MKtP *is called natural if for any input $x \in \{0,1\}^*$, the parameter of any query made by the reduction on x depends only on $|x|$.*

This notion of naturalness generalizes the notion used by Kabanets and Cai for m-reductions to Turing reductions. In fact, it is slightly weaker than their notion when restricted to m-reductions, since it does not constrain the length of the output, merely the parameter. As with the definition of parametric honesty, the condition on parameters is only required to hold when previous queries are answered correctly.

Naturalness is incomparable to parametric honesty - parametric honesty allows the parameter to vary but restricts its range of variation, while naturalness allows the parameter to have a large range of variation but insists that it is the same for all queries made on any input of a given length.

3 Parametric Honest Reductions

Our first result unconditionally rules out hardness of MKtP for E under parametric honest Turing reductions. For many results on hardness of MCSP and consequences thereof, MKtP is a 'test case' where analogous results can be shown unconditionally and where the ideas are often simpler [8, 16].

▶ **Theorem 7.** MKtP *is not hard for* E *under parametric honest Turing reductions.*

Proof. Let L be a tally set that is in E but not in P. The existence of such a tally set L follows by a standard diagonalization argument.

Now suppose, for the purpose of contradiction, that MKtP is hard for E under parametric honest Turing reductions. This implies that there is an oracle Turing machine M running in time at most cn^k, where c and $k \geq 1$ are constants, such that M decides L correctly with oracle access to MKtP, and moreover there is a constant $\epsilon > 0$ such that each query made by M on an input of length n has parameter at least n^ϵ. We show how to use M to decide $L \in$ P, contrary to the assumption on L.

Consider the following polynomial-time machine A that attempts to decide L. Given input 0^n, A runs the oracle machine M, answering each query made by M with 'yes'. When the simulation of M is complete, A accepts iff M accepts. Since M is polynomial-time, so is A. We show that A correctly decides L for large enough n.

Given input 0^n, M makes some sequence of oracle queries $q_1, q_2 \ldots q_t$ during the simulation by A, where $0 \leq t \leq cn^k$. We show that for each i, q_i has Kt complexity $O(\log(n))$. Observe that q_i can be generated by the universal machine U implicit in the definition of Kt complexity

in time poly(n) given descriptions of i, n and the oracle machine M: it simply needs to run M on 0^n answering the first $i-1$ queries made by M with 'yes', and then output the i'th query. The description length required is $O(\log(n))$ since $i \leq cn^k$, and moreover the time taken by U is polynomial in n, hence using the definition of Kt complexity, we have an $O(\log(n))$ upper bound on the Kt complexity of q_i. Let C be a constant such that the Kt complexity of each q_i is bounded above by $C\log(n)$.

By the assumption that the reduction is parametric honest, the parameter of q_i is at least n^ϵ, which for large enough n is greater than $C\log(n)$. Hence for large enough n and for each i, it follows that A's assumed answer for the oracle query q_i is correct for each i, and hence that A outputs the same answer as M would given the true oracle MKtP. Since M is an oracle Turing machine correctly implementing a reduction from L to MKtP, A is a polynomial-time Turing machine correctly deciding L for large enough n, contradicting the assumption on L.

◀

We now re-state and prove Theorem 1.

▶ **Theorem 8.** *If* MCSP *is* NP*-hard under parametric honest Turing reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

Proof. Suppose that MCSP is NP-hard under parametric honest Turing reductions. Let M be a polynomial-time oracle Turing reduction implementing a parametric honest reduction from SAT to MCSP, and let ϵ be a constant such that the parameter of any query made by M on any SAT instance of length x is bounded below by $|x|^\epsilon$.

Consider the following polynomial-time machine A which attempts to decide SAT. On an input x of SAT, A runs the oracle Turing machine M. Assume wlog that all queries asked by M have length a power of two - any queries for which this is not the case may be eliminated by assuming the answer is 'no'. For every query asked by M with length a power of two, A assumes the answer is 'yes', and continues the simulation of M. When the simulation concludes, A accepts iff M accepts.

Clearly A runs in polynomial time. If A decides SAT correctly, then we have that $\mathsf{NP} = \mathsf{P}$. Hence the Polynomial Hierarchy collapses to P. But this implies that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$; if the contrary were true, then E would collapse to the Polynomial Hierarchy using the Karp-Lipton theorem for E, and hence to P, contradicting the hierarchy theorem for deterministic time.

Suppose now that $\mathsf{NP} \neq \mathsf{P}$. In particular, this means that there are infinitely many instances on which A does not decide SAT correctly. If not, we could hardcode the finitely many instances on which A fails to compute SAT into a new polynomial-time machine A' which runs in polynomial time and solves SAT correctly on all instances, contradicting the assumption that $\mathsf{NP} \neq \mathsf{P}$.

Now, we can use the main result of Gutfreund, Shaltiel and Ta-Shma. They show that if $\mathsf{NP} \neq \mathsf{P}$, then for any polynomial-time algorithm A that fails to compute SAT, there is a polynomial-time Turing machine B and a constant d, such on input 1^n, B outputs three instances x_1, x_2, x_3 each of length n or length n^d such that for infinitely many n, A fails to decide SAT correctly on at least one of these three instances.

We show how to use B to argue again that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$. Consider the three instances x_1, x_2, x_3 output by B on input 1^n. For each $i, 1 \leq i \leq 3$, let Q_i be the concatenation of all queries made by M when M is simulated by A on input x_i. Let Y be the concatenation of Q_1, Q_2, Q_3. Note that $|Y| = \text{poly}(n)$, since M is a polynomial-time oracle Turing machine. Also note that each query made by M on an input of length n has parameter at least n^ϵ, by the parametric honesty of the reduction, as long as all previous queries were answered

correctly. Each such query was assumed by A to have a 'yes' answer. It could not be the case that all such answers were correct for all queries made by M on inputs x_1, x_2, x_3 - if they were, then A would correctly solve SAT on x_1, x_2, x_3, using the fact that M implements a correct polynomial-time Turing reduction from SAT.

Thus we get that there is a first query made by M on one of x_1, x_2, x_3 that has parameter at least n^ϵ and is wrongly answered 'yes'. Fix such a query y. Since M implements a Turing reduction from SAT to MCSP, y is the truth table of a Boolean function on $O(\log(n))$ bits that requires circuits of size n^ϵ. We have that y is a substring of Y, hence Y must also require circuits of size $\Omega(n^\epsilon)$. We are using here the simple fact that if y is a substring (with length a power of two) of the truth table Y of a Boolean function with circuits of size s, then y is the truth table of a Boolean function with circuits of size $O(s)$.

Hence, for infinitely many n, on input 1^n, we can compute the truth table Y of a Boolean function on $O(\log(n))$ bits such that the function requires circuits of size n^ϵ, i.e., of exponental size in the length of the input to the Boolean function. This implies $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$. ◀

4 Natural Reductions

We require a technical lemma allowing us to use indirect diagonalization based on robustly-often simulations.

▶ **Lemma 9.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function such that $T(\mathrm{poly}(T(\mathrm{poly}(n)))) = o(2^n)$. If SAT is $T(\mathrm{poly}(n))$-robustly often in time $T(n)$, then $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.*

Proof. Suppose SAT is $T(\mathrm{poly}(n))$-robustly often in time $T(n)$. By Proposition 3, we have that $\Sigma_2\mathsf{P}$ is robustly often in time $T(\mathrm{poly}(T(\mathrm{poly}(n))))$. Assume for the sake of contradiction that $\mathsf{E} \subseteq \mathsf{SIZE}(\mathrm{poly})$. The Karp-Lipton theorem for E, credited to Meyer [19], implies that E collapses to $\Sigma_2\mathsf{P}$. Note that if T is time-constructible, $T' = T(\mathrm{poly}(T(\mathrm{poly}(n))))$ is also time-constructible. Thus we have that E is robustly often in time T' for a time-constructible $T' = o(2^n)$, contradicting Proposition 4. ◀

We now re-state and prove Theorem 2. In the proof below, we use the fact that SAT is paddable - there is a polynomial-time computable *padding* function f such that for each $x \in \{0,1\}^*$, and for each $m \geq |x|$, $|f(x, 1^m)| = m$ and $f(x, 1^m)$ is in SAT iff x is in SAT. Of course, this depends on the encoding of SAT, but it suffices for us that there is some encoding for which paddability is true in the form above, and all other standard properties of SAT continue to hold.

▶ **Theorem 10.** *If MCSP is NP-hard under natural Turing-reductions, then $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$*

Proof. Suppose that MCSP is NP-hard under natural Turing reductions. Let M be an oracle Turing machine implementing a natural reduction from SAT to MCSP. By the definition of natural reduction, this means that there is a function $s : \mathbb{N} \to \mathbb{N}$ such that all queries of M on input of size n for SAT have parameter $s(n)$. If there are no queries that M makes on an input of size n, we set $s(n) = 0$.

Let $s_1(n) = \log(n)^{\log\log(n)}$, and $s_2(n) = \log(n)^{\log\log\log(n)}$. Moreover let $r(n) = n^{\log(n)}$. We define two Turing machines M_1 and M_2 that attempt to decide SAT using the oracle machine M.

Machine M_1 operates as follows. On input x of size n, M_1 computes strings $x_1 \ldots x_{r(n)-n}$, where $x_i = f(x, 1^{n+i})$. Here f is the padding function for SAT. Let $x_0 = x$. M_1 iteratively does the following starting with $i = 0$. It simulates M on x_i. If an oracle query is made with parameter k, it checks if k is at most $s_1(n+i)$. Note that since M implements a natural

reduction, $k = s(n+i)$. If k is at most $s_1(n+i)$, it solves the corresponding MCSP instance by brute force search in time at most $\text{poly}(n+i)2^{s_1(n+i)^2}$ and continues the simulation. Note that any succeeding queries must also have parameter at most $s_1(n+i)$, therefore the simulation continues to completion. M_1 accepts iff M accepts. If no oracle queries are made, then too the simulation continues to completion, and M_1 returns the same answer as M.

In case the parameter k for a query is greater than $s_1(n+i)$, M_1 increments i, and repeats the process, as long as $i \leq r(n) - n$. If $i > r(n) - n$, M_1 simply runs the simulation of M on x to completion, now answering all queries by brute-force search regardless of the parameter size.

Claim: Either SAT is decidable in time $\text{poly}(r(n))2^{s_1(r(n))^2}$, or there is an infinite sequence of disjoint intervals $I_j = [n_j, n'_j)$ of input lengths such that $n'_j = r(n_j)$ for each j, and moreover for each j, if $n \in I_j$, then $s(n) > s_1(n)$.

We establish the Claim. Suppose that SAT is not decidable in time $\text{poly}(r(n))2^{s_1(r(n))^2}$. We argue that for any $n_0 \in \mathbb{N}$, there exists an input length $n > n_0$ such that $s(n) \geq s_1(n)$, and moreover $s(m) \geq s_1(m)$ for every $n < m < r(n)$. Indeed, if this were not the case, for every $n > n_0$ one of the iterations of M_1 would succeed for some $i < r(n) - n$, and this would mean that the entire simulation would complete in time at most $\text{poly}(r(n))2^{s_1(r(n))^2}$. Note that when the simulation completes, it does solve SAT correctly, as M is an oracle Turing reduction correctly reducing SAT to MCSP, and any completing simulation uses correct answers to oracle queries.

Now we pick the intervals I_j inductively as follows. Let n_1 be the least integer such that $s(m) > s_1(m)$ for every $n_1 \leq m \leq r(n_1)$, and let $n'_1 = r(n_1)$. Inductively, suppose we have picked $n_1 \ldots n_j$. We pick n_{j+1} to be the least integer such that $n_{j+1} > r(n_j)$, and moreover $s(m) > s_1(m)$ for each $n_{j+1} \leq m \leq r(n_{j+1})$. Such an integer exists by the argument of the previous paragraph. Let $n'_{j+1} = r(n_j)$, and set $I_j = [n_j, n'_j)$. This sequence of intervals is clearly disjoint, and satisfies the property in the Claim.

The first disjunct of the Claim implies that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$, using Lemma 9 and the fact that $T(n) = \text{poly}(r(n))n^{s_1(r(n))}$ is time-constructible and satisfies $T(\text{poly}(T(\text{poly}(n)))) = o(2^n)$, for the given choice of r and s_1. Hence, in the rest of our proof, we assume that the second disjunct holds, i.e., that there is an infinite sequence of long intervals within which each input length generates queries with a large parameter.

Now we define machine M_2. On input x of length n, M_2 uses M to try to solve the search version of SAT, i.e., to find a satisfying assignment for x if one exists. M_2 uses the standard search-to-decision reduction for SAT based on self-reducibility, simulating M to answer any decision questions as described below. In order to find a satisfying assignment for x, this search-to-decision procedure might call the decision procedure for SAT on inputs x' such that $|x'| < |x|$; in such cases, M_2 pads up x' to length $|x|$ by using the padding function $f(x', 1^{|x|})$ and runs the simulation of M on $f(x', 1^{|x|})$. This guarantees that all inputs on which M is run during the simulation have the same length. At the end of the search-to-decision procedure, if a satisfying assignment is successfully found, M_2 accepts, else it rejects.

Next we describe how M_2 uses M to decide if an input is in SAT. M_2 simulates the oracle machine M on the input. If M asks a query with parameter greater than $s_2(n)$, M_2 assumes the answer is 'yes' and continues the simulation. If M asks a query with parameter at most $s_2(n)$, M_2 solves the query by brute force search and continues the simulation. At the end of the simulation, M_2 accepts iff M accepts.

M_2 always halts within time $\text{poly}(n)2^{s_2(n)^2}$. Unlike machine M_1, machine M_2 is not guaranteed to solve SAT correctly. However, since M_2 has been modified to solve the search version of SAT, the only way M_2 can make a mistake on x is to answer 'no' when the true

answer is 'yes'. We will argue that whether or not M_2 solves SAT correctly, circuit lower bounds follow.

We consider two cases. The first is that there is a $2^{s_2(n)^3}$-robust set S on which M_2 solves SAT correctly. In this case, we can apply Lemma 9 with $T(n) = \text{poly}(n)2^{s_2(n)^2}$, noting that T is time-constructible and that $T(\text{poly}(n)) < 2^{s_2(n)^3}$ for large enough n. Thus we get that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$ in this case.

The remaining case is that there is no $2^{s_2(n)^3}$-robust set S on which M_2 solves SAT correctly. In this case, we apply the main idea of [12], using in addition our assumption that there is an infinite sequence $\{I_j\}$ of disjoint intervals $[n_j, r(n_j))$ of input lengths such that for each j and $n \in [n_j, r(n_j))$, we have that $s(n) > s_1(n)$.

If there is no $2^{s_2(n)^3}$-robust set S on which M_2 solves SAT correctly, there must be a constant k such that for each n, there is $m < 2^{ks_2(n)^3}$ such that M_2 fails to solve SAT correctly on some input of length m. Now we use the existence of the sequence $\{I_j\}$ of intervals from the Claim. Since $r(n) = n^{\log(n)}$ is significantly larger than $2^{ks_2(n)^3}$ for large enough n, it follows that there is an infinite sequence of input lengths $\{m_i\}$ such that the following hold: (i) M_2 fails to solve SAT correctly on some input x of length m_i such that M on input x asks a query with parameter at least $s_1(m_i)$ (ii) For all input lengths m such that $m_i \leq m \leq 2^{s_2(m_i)^4}$, $s(m) \geq s_1(m)$.

Now we apply the main idea of [12], by attempting to use the simulation M_2 itself to find a small set of inputs for which M_2 fails to decide SAT correctly on at least one of these inputs. We define a Turing machine A as follows. On input 1^m, A formulates the question of whether there is an instance of size m on which M_2 fails to solve SAT correctly and asks a query with parameter at least $s_2(m)$ as an instance y of SAT. Since M_2 runs in time $\text{poly}(n)2^{s_2(n)^2}$, such an instance y of size at most $2^{s_2(m)^3}$ for large enough m can be computed in time at most $2^{s_2(m)^3}$. A then runs M_2 on y to find a satisfying assignment for y, as in the proof of the main result of [12]. For all $m = m_i$ for some i, either A succeeds, in which case it finds an instance x of length m on which M_2 fails, or it finds a set of at most three instances y_1, y_2, y_3 such that $|y_j| = |y|$ for each $1 \leq j \leq 3$, and M_2 fails on at least one of these three instances. Note that by item (ii) in the para above, in the first case x must ask some query with parameter at least $s_1(m)$ that is answered wrongly, and in the second case, one of y_1, y_2, y_3 must ask some query with parameter at least $s_1(|y|)$ that is answered wrongly. In either case, a wrongly answered query must be the truth table of a hard function. In the first case, we have that the concatenation Z_i of all queries asked by x is the truth table of a function on $O(\log(m))$ input bits that does not have circuits of size $s_1(m)$, and in the second case we have that the concatenation Z_i of all queries asked by y_1, y_2, y_3 must be the truth table of a function on $O(\log(|y|))$ input bits that does not have circuits of size $s_1(|y|)$. A outputs Z_i.

In either case, A runs in time $\text{poly}(|Z_i|)2^{s_2(|Z_i|)^3}$ and outputs a string Z_i that does not have circuits of size $s_1(|Z_i|)$. Since $s_2(|Z|) = s_1(|Z|)^{o(1)}$, this implies that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$. ◀

5 Open Problems

The main open problem is to derive circuit lower bounds from unrestricted Turing reductions from SAT to MCSP. One obstacle here is that we don't we know that MKtP is not hard for E under Turing reductions - this is a potentially simpler 'test case' that could be indicative. We note that [16] have shown that a gap version of MKtP is not hard for E, when the gap between the upper and lower thresholds for Kt-complexity is $\omega(\log(n))$.

An easier problem is to derive circuit lower bounds from unrestricted many-one reductions

from SAT to MCSP. Even this is unknown, though we do know that such reductions imply EXP $\neq$ ZPP [21, 17].

References

1. Eric Allender. The complexity of complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017.
2. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.
3. Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 25–32, 2014.
4. Eric Allender, Joshua A. Grochow, and Cristopher Moore. Graph isomorphism and circuit size. *CoRR*, abs/1511.08189, 2015.
5. Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing DNF formulas and AC0 circuits given a truth table. *Electronic Colloquium on Computational Complexity (ECCC)*, (126), 2005.
6. Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 54:1–54:14, 2017.
7. Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. In *International Symposium on Theoretical Aspects of Computer Science* (STACS), pages 21–33, 2015.
8. Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011.
9. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1st edition, 2009.
10. Sebastian Czort. The complexity of minimizing disjunctive normal form formulas. Master's Thesis, University of Aarhus, 1999.
11. Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Information and Computation*, 256:149–159, 2017.
12. Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Computational Complexity*, 16(4):412–441, 2007.
13. Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, pages 285–306, 1965.
14. Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *33rd Computational Complexity Conference, CCC 2018*, pages 5:1–5:31, 2018.
15. Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017.
16. Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Conference on Computational Complexity* (CCC), pages 18:1–18:20, 2016.
17. John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science* (FSTTCS), pages 236–245, 2015.
18. Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.
19. Richard Karp and Richard Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 302–309, 1980.
20. William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

21 Cody Murray and Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

22 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

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

24 Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.