Easiness Amplification and Uniform Circuit Lower Bounds*

Cody D. Murray[1] and R. Ryan Williams[2]

1 **CSAIL and EECS, MIT, Cambridge, MA, USA**
`cdmurray@mit.edu`
2 **CSAIL and EECS, MIT, Cambridge, MA, USA**
`rrw@mit.edu`

Abstract

We present new consequences of the assumption that time-bounded algorithms can be "compressed" with non-uniform circuits. Our main contribution is an "easiness amplification" lemma for circuits. One instantiation of the lemma says: if $n^{1+\varepsilon}$-time, $\tilde{O}(n)$-space computations have $n^{1+o(1)}$ size (non-uniform) circuits for some $\varepsilon > 0$, then every problem solvable in *polynomial* time and $\tilde{O}(n)$ space has $n^{1+o(1)}$ size (non-uniform) circuits as well. This amplification has several consequences:

- **An easy problem without small LOGSPACE-uniform circuits.** For all $\varepsilon > 0$, we give a natural decision problem General Circuit n^{ε}-Composition that is solvable in $n^{1+\varepsilon}$ time, but we prove that polynomial-time and logarithmic-space preprocessing cannot produce $n^{1+o(1)}$-size circuits for the problem. This shows that there are problems solvable in $n^{1+\varepsilon}$ time which are not in LOGSPACE-uniform $n^{1+o(1)}$ size, the first result of its kind. We show that our lower bound is non-relativizing, by exhibiting an oracle relative to which the result is false.
- **Problems without low-depth LOGSPACE-uniform circuits.** For all $\varepsilon > 0$, $1 < d < 2$, and $e < d$ we give another natural circuit composition problem computable in $\tilde{O}(n^{1+\varepsilon})$ time, or in $O((\log n)^d)$ space (though not necessarily simultaneously) that we prove does not have SPACE$[(\log n)^e]$-uniform circuits of $\tilde{O}(n)$ size and $O((\log n)^e)$ depth. We also show SAT does not have circuits of $\tilde{O}(n)$ size and $\log^{2-o(1)} n$ depth that can be constructed in $\log^{2-o(1)} n$ space.
- **A strong circuit complexity amplification.** For every $\varepsilon > 0$, we give a natural problem Circuit n^{ε}-Composition and show that if it has $\tilde{O}(n)$-size circuits (uniform or not), then every problem solvable in $2^{O(n)}$ time and $2^{O(\sqrt{n \log n})}$ space (simultaneously) has $2^{O(\sqrt{n \log n})}$-size circuits (uniform or not). We also show the same consequence holds assuming SAT has $\tilde{O}(n)$-size circuits.

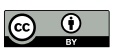
  As a corollary, if $n^{1.1}$ time computations (or $O(n)$ nondeterministic time computations) have $\tilde{O}(n)$-size circuits, then *all* problems in exponential time and subexponential space (such as quantified Boolean formulas) have *significantly* subexponential-size circuits. This is a new connection between the relative circuit complexities of easy and hard problems.

1998 ACM Subject Classification F.1.3 Relations Among Complexity Classes

Keywords and phrases uniform circuit complexity, time complexity, space complexity, non-relativizing, amplification

Digital Object Identifier 10.4230/LIPIcs.CCC.2017.8

* Supported by NSF CCF-1552651 (CAREER). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.



32nd Computational Complexity Conference (CCC 2017).
Editor: Ryan O'Donnell; Article No. 8; pp. 8:1–8:21

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Boolean circuit complexity and machine-based computation are nicely bridged by the notion of *uniform circuits*. These are circuit families where any particular circuit in the family can be efficiently generated on demand, by an algorithm consuming few resources. We use $\mathcal{C}$-uniform $\mathsf{SIZE}[s(n)]$ to (informally) denote the class of problems computable by a circuit family $\{C_n\}$ of $s(n)$-size such that the description of C_n is computable in $\mathcal{C}$ for all n. (The choice of circuit description/encoding can vary, and the results of our paper do not depend on these choices. See Section 2 for formal definitions.) Borrowing some terminology from data structures, one can say that uniform circuits naturally capture preprocessing/query tradeoffs in a static model: putting a problem in $\mathcal{C}$-uniform $\mathsf{SIZE}[s(n)]$ means that we can preprocess with $\mathcal{C}$ resources to build a size-$s(n)$ circuit, so that every subsequent n-bit instance of the problem can be computed (queried) with that circuit.

Most work on uniform circuit complexity over the last several decades has studied either the case where the class $\mathcal{C}$ is extremely weak, or is extremely strong. The extremely weak uniform models, such as $\mathsf{LOGTIME}$-*uniform circuits*, are the closest to machine-based computation: any particular wire or gate of a size-s circuit C_n in the family can be computed in only $O(\log s)$ time, proportional to the sizes of pointers to the wires/gates. The measures of "$\mathsf{LOGTIME}$-uniform circuit size" and "time" are known to coincide up to polylogarithmic factors [22], hence there are problems solvable in $n^{1.002}$ that are not in $\mathsf{LOGTIME}$-uniform $\mathsf{SIZE}[n^{1.001}]$, by the time hierarchy theorem [13, 33]. On the other end of the uniformity spectrum, *non-uniform* circuits require no computable bounds on generating the circuits, and far less is known: for example, no functions in huge classes like $\mathsf{TIME}[2^{O(n)}]^{SAT}$ are known to require even $4n$-size non-uniform circuits over the basis of all Boolean functions on two inputs, although some progress has recently been made on this front [9].

A few years ago, Santhanam and Williams [25] studied so-called "medium-uniform" circuits, where the complexity of generating a circuit is neither very weak nor very strong, e.g., $\mathsf{LOGSPACE}$, P, and P^{NP}-uniformity. Among other results, Santhanam and Williams [25] proved that for some k, there is a problem solvable in n^k time that does not have P-*uniform* linear-size circuits; that is, they proved $\mathsf{P} \not\subset \mathsf{P}$-uniform $\mathsf{SIZE}[O(n)]$. (In fact, they proved the stronger result that for every c, there is a k_c and a problem in $\mathsf{TIME}[n^{k_c}]$ that does not have P-uniform n^c-size circuits.) That is, they prove a *super-polynomial* time lower bound on preprocessing linear-size circuits for solving a problem in P. (It is believed there are problems in P without linear-size circuits *period*, no matter how much preprocessing is used, but this is an infamously difficult and famous open problem.) Their techniques led to similar results for $\mathsf{LOGSPACE}$-uniform branching programs, and lower bounds for NP versus $\mathsf{P}^{\mathsf{NP}}_{||}$-uniform linear size circuits.

There are two major drawbacks of the lower bound method of Santhanam and Williams. The first is its extreme non-constructivity: their method is a (very) indirect diagonalization argument. No particular problem in n^k time is known to exhibit the circuit size lower bound, and in fact the proof provides *no explicit bound on* k. That is, we cannot point to any problem in P satisfying the lower bound; we cannot even point to an upper bound on the time complexity of such a problem. This non-constructive phenomenon holds for all lower bounds in their work, creating a frustrating state of affairs. The second more serious drawback of their method is that it *relativizes*, which implies that there are hard barriers to what it can possibly prove. In particular, we cannot expect to prove results like $\mathsf{P} \not\subset \mathsf{SIZE}[O(n)]$ via such techniques.

1.1 Our Results

We introduce a non-relativizing method for exploiting the weakness of small-space computations, and for "amplifying" assumptions on the circuit complexity of easy problems. Using this method, we identify natural circuit composition problems which can easily be solved in low-polynomial time, yet we can prove non-trivial LOGSPACE-uniform circuit lower bounds for computing them. That is, *no* small-space algorithm can generate small circuits for solving the problems, despite their tractability. Our techniques give new insight into how to prove limitations on small-space computation. It is possible that similar ideas could potentially apply to non-uniform models of small-space computation, such as branching programs.

▶ **Definition 1.** In the $k(n)$-IO Circuit t-Composition problem, we are given a Boolean circuit C over AND/OR/NOT of size n with $k(n)$ inputs and $k(n)$ outputs, an input $x \in \{0,1\}^{k(n)}$, and integer $t \geq 1$. The task is to output

$$C^t(x) := \underbrace{(C \circ \cdots \circ C)}_{t}(x),$$

i.e., C composed for t times on the input x.

The $k(n)$-IO Circuit t-Composition problem can also be expressed as a decision problem, by including an index $i = 1, \ldots, k(n)$ as input, and outputting the ith bit of $C^t(x)$.

When $k(n) = n^{o(1)}$, we simply call the problem Circuit t-Composition.

Observe that $k(n)$-IO Circuit t-Composition can be easily solved in $\tilde{O}(n \cdot t)$ time and $\tilde{O}(n)$ space, by straightforward simulation of the given size-n circuit for t times. (As is standard, we let $\tilde{O}(t(n))$ denote $t(n) \cdot (\log t(n))^c$ for an unspecified constant $c > 0$.)

The circuit composition problem defined above is a sequential "chain" of circuit evaluations. A more general version of the composition problem, which we call General Circuit t-Composition (defined in the Preliminaries), permits connections between multiple inputs and outputs of a given circuit; Circuit t-Composition is a special case of it. General Circuit t-Composition is also solvable in $\tilde{O}(n \cdot t)$ time (see Section 2); it also requires $\tilde{\Omega}(n \cdot t)$ time to be solved (see Theorem 14). (We let $\tilde{\Omega}(s(n))$ denote $s(n)/(\log s(n))^c$ for a constant $c > 0$.)

LOGSPACE-Uniform Circuit Lower Bounds

First, we prove circuit lower bounds for generically composing n^ε copies of a circuit, which is an $\tilde{O}(n^{1+\varepsilon})$-time task:

▶ **Theorem 2.** *For all $\varepsilon \in (0,1)$, General Circuit n^ε-Composition does not have $n^{1+o(1)}$-size circuits constructible in logarithmic space.*

We stress that Theorem 2 does *not* relativize. In Appendix A, we exhibit (for every constant $k \geq 1$) an oracle A such that every language in $\mathsf{TIME}(n^k)^A$ has $O(n)$-size A-oracle circuits constructible in $\mathsf{LOGSPACE}^A$.

It is well known that every problem solvable in t time *does* have LOGTIME-uniform circuits of $O(t \log t)$ size [22], and that there are problems solvable in t time that do not have LOGTIME-uniform $O(t/\log^3 t)$-size circuits, by the time hierarchy theorem [13, 14]. Theorem 2 is a significantly stronger lower bound than what is provided by the time hierarchy: it shows that *arbitrary* logarithmic space preprocessing (running in, say, $n^{10^{10^{10}}}$ time) is not enough to reduce the resources needed to solve Circuit n^ε-Composition even slightly less

than $\tilde{O}(n^{1+\varepsilon})$. That is, LOGSPACE-uniform circuits cannot be made noticeably smaller than the time-bounded computations they may simulate, in general.

Theorem 2 is interesting not only because it does not relativize, but because there are few non-trivial lower bounds known against LOGSPACE, even as a uniformity condition. For random-access models, it is known that SAT is not solvable in (simultaneous) $n^{1.8}$ time and $n^{o(1)}$ space, but there are concrete limitations on known proof methods [11, 30, 6]. Slightly non-linear time lower bounds (for $n^{1-\Omega(1)}$ space) are known for some functions in P [1, 5]. Fortnow [10] proved (along with follow-up work by [3, 28]) that SAT does not have LOGSPACE-uniform $n^{1+o(1)}$-size $O(\log n)$-depth circuits. Santhanam and Williams [25] proved there is a language in LOGSPACE which does not have LOGSPACE-uniform branching programs of $O(n^k)$ size for every k, but (as mentioned earlier) the argument yields no explicit language and no explicit resource bounds on the language. (On the other hand, small branching programs seem to be a weaker model than small $O(\log n)$-depth circuits.)

Super-Logarithmic Space Lower Bounds

Turning to the more restricted model of polylog-depth circuits, we can obtain stronger lower bounds than prior work. Informally, we define the problem $d(n)$-Depth Circuit t-Composition analogously to Circuit t-Composition, except the inputs are restricted to circuits of $d(n)$ depth and $d(n)$ input/output bits. The $d(n)$-Depth Circuit t-Composition problem can be solved in about $n \cdot t$ time (like Circuit t-Composition) or in about $d(n) \cdot t$ space. We note that it is open whether $d(n)$-Depth Circuit t-Composition can be solved in polynomial time and $d(n)^c$ space simultaneously for $d(n) > \omega(\log n)$; this is the heart of the NC versus SC question [7, 21].

▶ **Theorem 3.** *For every $\varepsilon \in (0,1)$, $c \geq 1$, $d \in (1,2)$, and $d' < d$, the problem $(\log n)^d$-Depth Circuit n^ε-Composition does not have $\mathsf{SPACE}[(\log n)^{d'}]$-uniform circuits of $n \cdot (\log n)^c$ size and $O((\log n)^{d'})$ depth.*[1]

In other words, we have a problem in $\mathsf{TIME}[n^{1+\varepsilon}] \cap \mathsf{SPACE}[\log^{2-\varepsilon/2}]$ that does not have $\tilde{O}(n)$-size $O(\log^{2-\varepsilon} n)$-depth circuits constructible in $n^{O(\log^{1-\varepsilon} n)}$ time and $O(\log^{2-\varepsilon} n)$ space, for every $\varepsilon \in (0,1)$. Theorem 3 is a significant advance over prior results in the area, which could only prove lower bounds of this form against NP-hard and coNP-hard problems such as SAT and $\overline{\text{SAT}}$ [10, 3], or against non-explicitly given problems in NC [25]. We stress that $O(n)$-size $O(\log n)$-depth lower bounds are in general far more difficult to reason about than one might think: it is open whether every language in $\mathsf{NTIME}[2^n]$ has non-uniform $O(n)$-size $O(\log n)$-depth circuits.

"Easiness Amplification" for Small Circuits

While the problem of proving P does not have $O(n)$-size circuits is notoriously hard, one may try assuming that $\mathsf{P} \subset \mathsf{SIZE}(O(n))$, and record absurd conclusions that follow from it. Might we be able to reach something so absurd that it is provably contradictory? A string of work [17, 12, 25, 8] has established consequences of this form.

---

[1] One might initially believe that Theorem 3 follows quickly from the space hierarchy [26]. It does not: recall the problem $(\log n)^{2-o(1)}$-Depth Circuit n^ε-Composition being lower-bounded is solvable in $n^{1+\varepsilon}$ time, yet we are proving nearly-quadratic space lower bounds for constructing nearly-quadratic depth circuits for it.

The key to the above lower bounds is a lemma which demonstrates how small circuits for these composition problems can be applied to construct small circuits for many more (presumably harder) problems. Let $\mathsf{TISP}[t(n), s(n)]$ denote the class of languages decidable in $t(n)$ time and $s(n)$ space (simultaneously).

▶ **Lemma 4** (Easiness Amplification). *Let $\epsilon > 0$ and let $s(n) \leq \tilde{O}(n)$.*

- *If $s(n)$-IO Circuit n^{ε}-Composition has $\tilde{O}(n)$ size circuits, then every problem in $\mathsf{TISP}[n^{k(n)}, s(n)]$ has $n \cdot (\log n)^{O(k(n)/\epsilon)}$ size circuits, for all constructible functions $k(n) \leq O(\log n / \log\log n)$.*[2]
- *If $s(n)$-IO Circuit n^{ε}-Composition has $n^{1+o(1)}$ size circuits, then for every constant $k \geq 1$, every problem in $\mathsf{TISP}[n^k, s(n)]$ has $n^{1+o(1)}$-size circuits.*

That is, assuming a single problem (solvable in $\tilde{O}(n^{1+\varepsilon})$ time) has nearly-linear-size circuits, we prove that a *much larger* class of problems also has very small circuits (note that $s(n)$-IO Circuit n^{ε}-Composition is in $\mathsf{TISP}[n^{1+\varepsilon}, s(n)]$). We call this phenomenon *easiness amplification*, in contrast with the study of hardness amplification. Let us carefully explain our choice of this term.

In *hardness amplification*, one shows that if a problem from a class $\mathcal{C}$ is "hard" in one sense, one can find another problem from $\mathcal{C}$ that is even "harder." That is, in a hardness amplification theorem, the class of problems being solved remains *about the same*, but the computational lower bound is strengthened in the conclusion (we are amplifying the hardness). However, in what we call easiness amplification, the computational model remains *about the same*, but the class of problems being solved is strictly *increased* in the conclusion: one strictly increases the set of problems which are shown to be easy. To give a specific example and a non-example of easiness amplification, we would say that $\mathsf{NP} \subset \mathsf{P/poly} \Rightarrow \mathsf{PH} \subset \mathsf{P/poly}$ is an easiness amplification, because the notion of "easy" in the conclusion is unchanged from that of the hypothesis. but the set of problems being solved has (probably) strictly increased in the conclusion. On the other hand, $\mathsf{NP} = \mathsf{P} \Rightarrow \mathsf{NEXP} = \mathsf{EXP}$ is not an easiness amplification: the computational model P "blows up" to the larger class EXP.

Lemma 4 says that if the above circuit composition problem (which is in $n^{1+\varepsilon}$ time and $\tilde{O}(n)$ space) had $\tilde{O}(n)$-size circuits, then *every* problem in $n^{o(\log n/\log\log n)}$ time and $\tilde{O}(n)$ space has $n^{1+o(1)}$ size circuits. That is, from a modest speed-up of the circuit composition problem with non-uniform circuits, one obtains an incredible non-uniform simulation of a much larger complexity class. The following is immediate from Lemma 4.

▶ **Corollary 5.** *If n-IO Circuit n^{ε}-Composition has $\tilde{O}(n)$ size circuits, then*

$$\mathsf{TISP}\left[n^{(\log n)/\log\log n}, \tilde{O}(n)\right] \subseteq \mathsf{SIZE}[O(n^2)].$$

By a standard padding argument, we also have $\mathsf{TISP}\left[2^n, 2^{\sqrt{n \log n}}\right] \subseteq \mathsf{SIZE}\left[2^{O(\sqrt{n \log n})}\right]$.

That is, nearly-linear size circuits for circuit composition imply polynomial-size circuits for some problems that are only known to be solvable in *super-polynomial* time, such as detecting a clique of $O(\log n / \log\log n)$ nodes. The second part of Corollary 5 shows that a small improvement on the circuit complexity of a problem solvable in $n^{1+\varepsilon}$ time implies truly *subexponential* circuit upper bounds on *all* problems solvable in 2^n time and $2^{\sqrt{n}}$

[2] As usual in machine-based complexity, one must worry if the functions under consideration are constructible within the resource bounds of the corresponding complexity classes. Throughout the paper, we say a function is "constructible" when it satisfies precisely that condition.

space. For example, the *quantified Boolean formula* problem on formulas of k variables and $2^{\sqrt{k \log k}} \cdot \text{poly}(k)$ size would have a circuit family of $2^{O(\sqrt{k \log k})}$ size. This consequence is in stark contrast to general beliefs regarding exponential time computation, e.g., the Exponential Time Hypothesis of Impagliazzo, Paturi, and Zane [16] which posits that 3-SAT does not have $2^{o(n)}$-time algorithms. Previously, it was only known that $\mathsf{P} \subset \mathsf{SIZE}[O(n)]$ implies $\mathsf{TIME}[2^{O(n)}] \subset \mathsf{SIZE}[2^{o(n)}]$ (by a simple padding argument).

It is interesting to contrast the circuit upper bound consequences of Corollary 5 with the fact that $\mathsf{P} \subset \mathsf{SIZE}[O(n)]$ also implies the negative consequence $\mathsf{P} \neq \mathsf{NP}$ [17]. It would be interesting if $\mathsf{NP} \subset \mathsf{SIZE}[O(n)]$ implied even smaller circuit constructions for PSPACE problems.

Circuit Lower Bounds for SAT

Some of our results extend to the (much harder) SAT problem; in that setting, we can use old proof techniques that are similar to those in prior work on SAT time-space tradeoffs. First we show (Section 4.1) that SAT requires superlogarithmic-space uniform circuits of $\tilde{O}(n)$ size and nearly $\log^2 n$ depth:

▶ **Theorem 6.** *For all $d < 2$, and all $c \geq 1$, SAT $\notin \mathsf{SPACE}[\log^d n]$-uniform $\mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^c, (\log n)^d]$.*

With respect to the depth measure, Theorem 6 is an improvement over previous uniform circuit lower bounds for SAT, which established $O(\log n)$-depth limitations (for bounded fan-in and "semi-unbounded" fan-in models). The proof uses the machinery of "alternation-trading proofs" for SAT lower bounds [10, 11, 30], where one assumes a very good SAT algorithm exists, and uses it along with known alternating "speed-up" theorems to derive a contradiction to a known result (generally, some time hierarchy theorem). We show how to use the assumed SAT circuits to obtain a contradiction to the *space* hierarchy theorem, rather than a time hierarchy. This alternate approach leads to stronger results.

Finally, we observe (via old ideas) that the easiness amplification results of Lemma 4 also hold for SAT:

▶ **Theorem 7.** *If SAT has $\tilde{O}(n)$-size circuits, then QBF has $2^{\tilde{O}(\sqrt{n})}$-size circuits.*

This is evidently a new connection between the relative circuit complexity of NP and PSPACE problems. The proof of this theorem can be found in Appendix B.

1.2 Intuition and Comparison

While our lower bounds do apply some ideas from prior work, the proofs of Theorem 2 and 3 have a particular inductive structure that is new to circuit lower bound proofs.[3] The key idea in our lower bounds is encapsulated by the following special case of Lemma 4:

▶ **Lemma 8** (Amplification From Circuit Composition: Special Case)**.** *For every $\varepsilon > 0$, if* Circuit n^{ε}-Composition *has $n^{1+o(1)}$ size circuits, then every problem solvable in* LOGSPACE *has $n^{1+o(1)}$ size circuits.*

Note there are no uniformity assumptions on the circuits in the lemma: the circuits may be arbitrary. This is a strong lower bound amplification result for a particular problem

[3] Readers who doubt this claim are invited to read the "Comparison With Prior Work" below.

in $n^{1+\varepsilon}$ time: small circuits for Circuit n^{ε}-Composition imply small circuits for all of LOGSPACE.

Let us sketch how the lemma is proved. Let M be a machine running in logarithmic space, and assume that small $n^{1+o(1)}$-size circuits exist for Circuit n^{ε}-Composition. Such circuits take two inputs: the description of a smaller circuit C, and an input y to C. We first set the input circuit C to be a $\tilde{O}(n)$-size circuit $C_0(x, \cdot)$ which simulates one step of M on an arbitrary input x of length n: treating the input y as an $O(\log n)$-bit configuration of M on x, $C_0(x, y)$ outputs the configuration y' corresponding to the next step of M on x. Then, $n^{1+o(1)}$-size circuits for Circuit n^{ε}-Composition can be used to construct $n^{1+o(1)}$-size circuits $\{C'_n\}$ which can simulate M on x for about n^{ε} steps, by composing C on the input y for n^{ε} times (using $O(\log n)$ copies, one for each bit of the output configuration).

Now suppose we feed the C'_n circuits as input to Circuit n^{ε}-Composition instead of the C_0 circuits, and repeat the above argument. Note that the C'_n's are only slightly larger than the C_0's. The circuits C'_n (which simulate n^{ε} steps) are being composed for n^{ε} times on the input y. Thus we obtain $n^{1+o(1)}$-size circuits $\{C''_n\}$ which can simulate arbitrary logspace computations for about $n^{2\varepsilon}$ steps. (Please note that this is not obviously true, and our wording here should be taken only as intuition. For example, we have not specified here how to handle arbitrary inputs x of length n.) Once we have the right setup, we can "repeat" the argument for $O(k/\varepsilon)$ times, each time with the new circuit family obtained from the previous iteration plugged in. From this it will follow that n^k-time log-space computations have $n^{1+o(1)}$-size circuits. In particular, given that small-space computations always have small configurations, and assuming we have nearly-linear size circuits that can simulate LOGSPACE for a moderate number of steps, we can concoct new circuits which can simulate LOGSPACE for an arbitrary polynomial number of steps, while keeping the circuit size around $n^{1+o(1)}$.

To prove the main circuit lower bound of Theorem 2, we start by assuming that General Circuit n^{ε}-Composition has logspace-uniform $n^{1+o(1)}$-size circuits, and wish to derive a contradiction. First we note that General Circuit n^{ε}-Composition is "complete" for $n^{1+\varepsilon}$ time in a certain precise sense, and thus cannot be solved faster than this bound. Using the argument of the above paragraph, we derive that every language in LOGSPACE also has almost-linear-size circuits. But if every language in LOGSPACE has small circuits, it can be shown that every logspace-uniform circuit family can be produced by *very small* circuits, by a "de-padding"' trick of Santhanam and Williams [25] which gives the uniform algorithm much smaller inputs. Indeed, assuming LOGSPACE has almost-linear-size circuits, it can be shown that every problem with small logspace-uniform circuits can also be decided very efficiently, with only $o(n)$ bits of advice. We can use this consequence to prove that General Circuit n^{ε}-Composition is solvable so efficiently that it contradicts our earlier time lower bound for the problem.

Note that if the original circuits for Circuit n^{ε}-Composition are assumed to be uniform (or of low depth, respectively) then the composition circuits in the above iterated constructed are also uniform (or of low depth, respectively). The main goal in the proof of the depth lower bound (Theorem 3) is to extend the above amplification lemma to simulate all problems in SPACE$[(\log n)^d]$ with uniform circuits of depth only $o(\log n)^d$. But such circuits can always be evaluated in SPACE$[o((\log n)^d)]$, so this consequence contradicts the space hierarchy.

We observe that (in contrast to the techniques of Santhanam and Williams) the proof technique in the amplification lemma looks to be inherently *non-relativizing*. The first circuit in our composition only simulates a logspace machine for one step, and our composition problem only simulates a given circuit for some n^{ε} steps, crucially using the fact that the output of the circuit (the configuration size) is only $n^{o(1)}$ bits at every stage of the induction.

Informally, there is no "room" in our circuit simulation of LOGSPACE to write down long oracle queries. To confirm this intuition, we construct an oracle relative to which our main lower bound is false, in Appendix A.

The analogous lower bound results for SAT follow from the fact that the circuit composition problem can also be solved using alternations rather than computing it serially: one can simply (existentially) guess the intermediate values output in the circuit composition, and (universally) verify the outputs in parallel. If SAT has nearly-linear-size circuits, then Σ_kSAT also has nearly-linear-size circuits, which also allows for a very efficient circuit simulation of small-space computation.

Comparison With Prior Arguments

We argue that the above proof technique (behind Lemma 4, Theorem 2, and Theorem 3) is a fundamentally different way to derive an efficient simulation of small-space computation, compared to earlier arguments.

1. **SAT Time-Space Lower Bounds.** In the SAT time-space lower bounds based on extending Savitch's theorem (such as Fortnow-Lipton-Van Melkebeek-Viglas [10, 11]), the use of *alternating computation* is critical: generalizing Savitch's Theorem, one constructs a simulation which "guesses" a few configurations of a small-space machine that will be visited later in the computation, then "universally verifies" the guesses independently. In this way, the simulation problem for a small-space computation is partitioned into small parts which can then be solved independently (in parallel). The proof of the Easiness Amplification Lemma (Lemma 4) uses no alternation or parallelism. Indeed, Lemma 4 shows how the *serial* process of simulating a small-space computation for t steps can be self-improved by assuming that $n^{1+\varepsilon}$ steps can be simulated with $n^{1+o(1)}$ -size circuits, and repeatedly feeding "small circuit copies" into circuits which perform this efficient step simulation. Note that in our later lower bounds for SAT, we do use alternations, showing that this methodology can derive similar results but under a seemingly stronger assumption (namely, the assumption "SAT has small circuits" instead of "circuit composition has small circuits").
2. **Hardness Amplification From Self-Reducibility.** The proof technique here is also different from earlier arguments based on the self-reducibility of a problem (such as those found in Lipton-Viglas, Allender-Koucky, and Lipton-Williams [18, 2, 19]). In those arguments, one decomposes a given computation into disjoint "parts", applies an assumed "good" simulation to each part separately, then proves that the new overall simulation is now similarly "good." The results proven in these papers are hardness amplifications: given a "pretty good" simulation of some particular kind of resource-bounded computation, we can obtain an extremely efficient simulation of the same bounded computation. (For example, if the NC^1-complete formula evaluation problem has TC^0 circuits of polynomial size, then it also has TC^0 circuits of $n^{1.0001}$ size [2]. Thus a weak TC^0 size lower bound against this NC^1 problem would separate NC^1 from TC^0.) Our results have a different character: we show that if one can simulate ultra-efficient computations with small circuits, then one can simulate all "pretty good" computations with small circuits. *We are improving the class of computations being simulated, instead of improving the simulation itself.* Again, this is the difference between hardness amplification and what we call "easiness amplification."
3. **Top-Down Versus Bottom-Up.** In Lemma 4, we begin with a small circuit that simulates one step of the computation correctly, and apply the hypothesis in a way that lets us build a (slightly larger) circuit simulating n^{ε} steps correctly. From that circuit, and our

hypothesis, we build another (slightly larger) circuit simulating $n^{2\varepsilon}$ steps correctly, and so on, until we have obtained a small circuit which can simulate n^k steps correctly for any desired k. Methodologically, this is a "bottom-up" way of simulating small-space computations faster: building up small circuits for simulating long running times, starting with a trivial circuit for simulating one step.
This should be contrasted with the "top-down" approach of extensions of Savitch's Theorem in item 1 above, where one guesses a way to partition the computation into pieces, then verifies the pieces recursively. The arguments based on self-reducibility in item 2 above also have a "top-down" form: they start by decomposing the entire computation into small parts, then substitute small copies of an improved circuit in place of each of the parts. Because we view the machine simulation problem in a bottom-up way, this allows us to prove a lower bound on a much simpler problem (a circuit composition problem solvable in $n^{1+\varepsilon}$ time) compared to earlier unconditional lower bounds of this type (for SAT and for QBF).

The most canonical results in the same spirit of our work are classical theorems such as $\mathsf{NP} \subset \mathsf{P/poly} \Rightarrow \mathsf{PH} \subset \mathsf{P/poly}$, where one replaces the quantifiers in a Σ_k computation by larger circuits. Of course there are other obvious differences there: in our setting, we want to keep the circuit size basically fixed (around $n^{1+o(1)}$) and we want to increase the *running times* of the problems we can solve with such circuits, in each inductive step. As far as we can tell, we need to be simulating small-space computations to pull off this kind of easiness amplification. It would be highly desirable to remove the small-space restriction in these lower bounds.

2 Preliminaries

We assume basic familiarity with concepts in complexity theory [4]. Below are some definitions and notions specific to this paper.

▶ **Definition 9.** A language L is in the class $\mathsf{TISP}[t(n), s(n)]$ if it can be decided in $O(t(n))$ time and $O(s(n))$ space simultaneously on a multitape Turing machine.

In our results on the SAT problem, we also use standard notions of alternating Turing machines:

▶ **Definition 10.** A language L is in the class $\Sigma_{a(n)}\mathsf{TISP}[t(n), s(n)]$ if it can be decided in $O(t(n))$ time and $O(s(n))$ space simultaneously on a multitape Turing machine using at most $a(n)$ alternations.

Sometimes we say "algorithm" instead of "multitape Turing machine." These should be thought of as synonymous. We need the multitape model to ensure that our algorithmic computational model has an efficient translation to small-size circuits.

In the following, let $\mathcal{C}$ be any time or space complexity class (such as $\mathsf{TIME}[n^2]$, P, $\mathsf{SPACE}[\log^2 n]$, etc.). For a circuit C, let $|C|$ be the length of its description in binary.

▶ **Definition 11.** A $\mathcal{C}$-uniform circuit family $\{C_n\}$ has the property that there is a multitape Turing machine A implementable in $\mathcal{C}$, such that for every n, $A(1^n)$ prints the $|C_n|$-length description of C_n in binary. (Strictly speaking, as the class $\mathcal{C}$ consists of decision problems, A should output only one bit. This is easily accommodated by requiring for all n and $i = 1, \ldots, |C_n|$ that $A(1^n, i)$ outputs the ith bit of the description of C_n.)

For example, a P-uniform circuit family comes with a polynomial-time multitape Turing machine A which on 1^n prints the nth circuit in the family.

For a language $L \subseteq \{0,1\}^\star$, define $L_n = L \cap \{0,1\}^n$ to be the strings in L of length n. A language L is in $\mathsf{SIZE}[s(n)]$ if for every $n \geq 0$, there is a circuit of at most $s(n)$ gates that computes L_n. L is in $\mathsf{DEPTH}[d(n)]$ if for every $n \geq 0$ there is a circuit that computes L_n such that the longest path from source to sink in this circuit has length at most $d(n)$. L is in $\mathsf{SIZE\text{-}DEPTH}[s(n), d(n)]$ if for every $n \geq 0$ there is a circuit with at most $s(n)$ gates computing L_n such that the longest path in the circuit has length at most $d(n)$. For example, the class NC^1 can be rewritten as $\mathsf{SIZE\text{-}DEPTH}[n^{O(1)}, O(\log n)]$. The following observation is useful in our depth lower bounds:

▶ **Proposition 12.** $\mathsf{SPACE}[s(n)]$-*uniform* $\mathsf{DEPTH}[s(n)] \subseteq \mathsf{SPACE}[s(n)]$.

Proof. For a circuit family $\{C_n\}$ that is $s(n)$-space uniform, on an n-bit input we can always use $O(s(n))$ space to generate any gate information of the circuit C_n necessary for a simulation. Because C_n also has $s(n)$ depth, we only require $O(s(n))$ additional space to simulate it (for a reference, see Vollmer [29]). ◀

Generalized Circuit Composition

In the following, let $Z : \{0,1\}^k \to \{0,1\}^k$ be a function that always returns 0^k.

▶ **Definition 13.** In the General Circuit t-Composition problem, we are given:

- A k-bit input x.
- A circuit C_{in} of size n over the basis $\{AND, OR, NOT\}$ implementing a function from $3k$ bits to k bits (where k can range up to n).
- A circuit C_{out} of size t over the basis $\{C_{in}, Z\}$, implementing a function from k bits to k bits. In particular, C_{out} is presented as a DAG, where every node has indegree 0, 1, or 3; every edge of C_{out} carries a k-bit string.
- An integer j in $1, \ldots, k$.

The input is accepted iff the jth bit printed by $C_{out}(x)$ is 1.

That is, every node in C_{out} of indegree 3 implements C_{in}, taking in $3k$ bits and outputting k bits.) We need indegree 3 in order to carry out a multitape TM simulation effectively.)

In our lower bound proofs, we only require two key properties of General Circuit t-Composition.

1. The first is that n-IO Circuit t-Composition for a circuit C is a special case of the General Circuit-t-Composition Problem. This is true because n-IO Circuit t-Composition corresponds to the case where C_{out} is simply a straight line of t copies of C_{in}, where $C_{in}(x, 0^k, 0^k) = C(x)$ for all x.
2. The second property is that General Circuit t-Composition is essentially complete for $\tilde{O}(nt)$ time, as the below theorem demonstrates.

▶ **Theorem 14.** *Let $L \in \mathsf{TIME}[n^{1+\varepsilon}]$. L can be reduced in $\tilde{O}(n)$ time to* General Circuit n^ε-Composition.

Proof (Sketch). Let M be multitape Turing machine for L. We follow the proof of the Size Reduction Lemma (Lemma 3.2) in Lipton and Williams [19], which shows how to "decompose" an arbitrary time $n^{1+\varepsilon}$ computation into circuits of size n^ε in which every gate takes a constant number of $O(n)$-bit "blocks" of input, simulates an $O(n)$-time machine M' on the blocks, and outputs an $O(n)$-bit block.

In particular, they convert M into an equivalent two-tape *oblivious* M' (where, for every n and input x of length n, the tape head movements of $M'(x)$ depend only on n). M' runs in $t = \tilde{O}(n^{1+\varepsilon})$ time, via the Hennie-Stearns simulation [14]. This oblivious two-tape simulation is polylog-time uniform, in that we can determine the head positions in any given step $i = 1, \ldots, t$ in $\text{poly}(\log t)$ time.

Next, M' is made *block-respecting* in the sense of Hopcroft-Paul-Valiant [15]. This is a machine M'' running in $t'(n) = O(t(n))$ time, whose computation can be neatly partitioned into $\frac{t'(n)}{b(n)}$ time blocks of $O(b(n))$ steps each, and each tape is partitioned into $O(\frac{t'(n)}{b(n)})$ tape blocks of $O(b(n))$ cells each, for any constructible $b(n) \leq t(n)$. We set $b(n) = O(n)$. For each time block, and each tape head, the head stays within exactly one tape block. Therefore each time block can be viewed as running on an input of $O(b(n))$ bits, and each block outputs $O(b(n))$ bits (the new content of those tape blocks). The Hopcroft-Paul-Valiant simulation maintains the obliviousness of M'.

For our generalized circuit composition instance, the circuit C_{in} simply simulates a time block of length $b(n)$, assuming the initial input is of length n. It takes $O(n)$ bits and outputs $O(n)$ bits, having $\tilde{O}(n)$ size. The circuit C_{out} connects these n^{ε} time blocks together: each gate of C_{out} implements a time block. C_{out} is built by determining the head movements of the oblivious M'' spaced n^{ε} steps apart, and wiring together time blocks that share common tape blocks (or adjacent time blocks that share adjacent tape blocks, depending on the head position). By design, the resulting circuit C_{out} is equivalent to the original Turing machine M on n-bit inputs. ◀

As a corollary, there is a constant $c > 0$ such that General Circuit n^{ε}-Composition needs at least $n^{1+\varepsilon}/(\log n)^c$ time, by the time hierarchy theorem.

2.1 Simulating Bounded Space

Many results we obtain stem from composing circuits that simulate space-bounded computations. The following notion is useful:

▶ **Definition 15.** Fix a machine M, and an input length n. The simulation machine $Sim_{t(n)}(x, c, i)$ takes a string x of length n, configuration c of M on x, and an index bit i, simulates $M(x)$ from configuration c for $t(n)$ steps, then outputs the ith bit of the resulting configuration c'.

For machines M running in space $s(n)$, $Sim_{t(n)}$ has circuits of size $\tilde{O}(t(n) \cdot (n + s(n)))$. In our amplification lemma (Lemma 4), we construct much more efficient circuits, assuming circuit composition has small circuits.

3 LOGSPACE-Uniform Circuit Lower Bounds

We begin this section with our amplification lemma, which is used to prove most of the following results.

▶ **Reminder of Lemma 4.** *Let $\epsilon > 0$ and let $s(n) \leq \tilde{O}(n)$.*

- *If $s(n)$-IO Circuit n^{ε}-Composition has $\tilde{O}(n)$ size circuits, then every problem in $\mathsf{TISP}[n^{k(n)}, s(n)]$ has $n \cdot (\log n)^{O(k(n)/\epsilon)}$ size circuits, for all constructible functions $k(n) \leq O(\log n / \log\log n)$.*
- *If $s(n)$-IO Circuit n^{ε}-Composition has $n^{1+o(1)}$ size circuits, then for every constant $k \geq 1$, every problem in $\mathsf{TISP}[n^k, s(n)]$ has $n^{1+o(1)}$-size circuits.*

Proof. We start by proving the first bullet. Assume $s(n)$-IO Circuit n^ε-Composition has $\tilde{O}(n)$ size circuits. Recall that $s(n)$-IO Circuit n^ε-Composition takes as input a circuit C with $s(n)$ inputs and $s(n)$ outputs, a string $y \in \{0,1\}^k$, and an index i, then prints the ith bit of the n^ε-fold composition of C on y.

By assumption, $s(n)$-IO Circuit n^ε-Composition has a circuit family $\{E_n\}$ of $O(n \cdot (\log n)^d)$ size.

Now take an arbitrary $L \in \mathsf{TISP}[n^{k(n)}, s(n)]$ for some function $k(n) \leq O(\log n / \log\log n)$, and let M be a machine recognizing L in $n^{k(n)}$ time and $s(n)$ space. We will show that L has circuits of size $n \cdot (\log n)^{O(k(n)/\varepsilon)}$, by constructing the circuit inductively.

For the base case, let $M_0(x, c, i) = Sim_1$ (from Definition 15) be a machine which simulates M on x from the configuration c for one step, then outputs the ith bit of the next configuration. This simulation can easily be done in linear time, by first looking up the input bit read by the input head, simulating one step of the computation, then outputting the ith bit of the resulting configuration. By the usual conversion of algorithms into circuits, there is a circuit family that is equivalent to Sim_1 with $n \cdot (\log n)^a$ size for some constant a. Let $C_0(x, c, i)$ denote a generic circuit from this family. We choose a convention for expressing the description of C_0 such that, given a bit string D which is the description of C_0, and given an n-bit input x, a description D_x of the circuit $C_0(x, \cdot)$ (i.e., the first n inputs of C_0 are filled in with the bits of x) is obtained by substituting the n bits of x into n particular bit positions $i_1, \ldots, i_n$ of the description D. Such a description only takes $O(z \log z)$ bits to describe, where z the circuit size.

Now fix $b \geq 0$. Suppose we have constructed a circuit $C_b(x, c, i) = Sim_{n^{\varepsilon \cdot b}}$ of size $n \cdot (\log n)^{a+b\cdot(d+1)}$ that simulates $n^{\varepsilon \cdot b}$ steps of the machine M on x of length n. Consider the circuit

$$C_{b+1}(x, c, i) := E_{n\cdot(\log n)^{a+b\cdot(d+1)+1}}(C_b(x, \cdot), c, i)\,.$$

(Note that by our convention for encoding circuits, $C_b(x, \cdot)$ should be construed as a bit string but with n bit positions that are unassigned free variables.) Since the language $s(n)$-IO Circuit n^ε-Composition simulates $n^{\varepsilon \cdot b}$ steps of M with each evaluation of the circuit C_b, and the circuit C_b is being composed for n^ε times, $C_{b+1}(x, c, i)$ simulates $n^{\varepsilon \cdot b} \cdot n^\varepsilon = n^{\varepsilon \cdot (b+1)}$ steps of M on x.

Furthermore, since the circuit C_b is of size $O(n \cdot (\log n)^{a+b\cdot(d+1)})$, the binary representation of C_b has length $\ell = O(n \cdot (\log n)^{a+b\cdot(d+1)+1})$. Therefore the input to the circuit C_{b+1} has length $O(\ell)$, and the size of the circuit C_{b+1} would then be of size

$$O(\ell \cdot (\log \ell)^d) \leq O(n \cdot (\log n)^{a+b\cdot(d+1)+1} \cdot (\log(n(\log n)^{a+b\cdot(d+1)+1})^d)).$$

For $b < \log n / \log\log n$, we have $n \log^{a+b\cdot d} n \leq n^{1+d} \log^a n$; in that case, the size bound can be simplified to

$$O(n \cdot (\log^{a+b\cdot(d+1)+1} n) \cdot (\log n)^d) = O(n \log^{a+b\cdot(d+1)+(d+1)} n) = O(n \log^{a+(b+1)\cdot(d+1)} n).$$

We have shown that given a circuit C_b of size $O(n \cdot (\log n)^{a+b\cdot(d+1)})$ that simulates $n^{\varepsilon \cdot b}$ steps of M on x, we can construct a circuit C_{b+1} of size $O(n \log^{a+(b+1)\cdot(d+1)})$ that simulates $n^{\varepsilon\cdot(b+1)}$ steps of M on x. Therefore for every $b \leq o(\log n / \log\log n)$, there is a circuit C_b of size $n \cdot (\log n)^{O(b)}$ that simulates $n^{\varepsilon \cdot b}$ steps of the space-$s(n)$ machine M. Setting $b = k(n)/\varepsilon$ and c to be the initial configuration of M on inputs of length n, we obtain a circuit $C_{k(n)/\varepsilon}$ of size $n \cdot (\log n)^{O(k(n)/\varepsilon)}$ that can simulate the entire computation of M and output the final configuration of $M(x)$, which can then be used to decide L. Since L was arbitrary, we

conclude that

$$\mathsf{TISP}[n^{k(n)}, s(n)] \subseteq \mathsf{SIZE}[n \cdot (\log n)^{O(k(n)/\epsilon)}],$$

which completes the proof of the first bullet.

To prove the second bullet, let $\varepsilon \in (0,1)$ and $k \geq 1$ be constant. We run the same argument on an arbitrary machine M using n^k-time and $s(n)$-space, but instead we assume there are $n^{1+1/f(n)}$-size circuits for the n^ε-composition problem, where $f(n)$ is an unbounded function. For every constant $b \geq 0$, tracking the growth of C_b in the above argument, we obtain a circuit $C_b(x,c,i)$ of size at most $n^{(1+1/f(n))^{db}}$ (for some universal constant $d > 0$) for simulating $n^{\varepsilon b}$ steps of M on an input of length n. By setting $b := k/\varepsilon$ as in the previous case, the resulting circuit C_b can then simulate M entirely on all inputs of length n. We can define an unbounded function $g : \mathbb{N} \to \mathbb{N}$ such that

$$(1 + 1/f(n))^{\log f(n)} \leq 1 + 1/g(n)$$

for all n. Then for all constants b and for all sufficiently large n, the size of C_b is $n^{(1+1/f(n))^{db}} \leq n^{(1+1/f(n))^{\log f(n)}} \leq n^{1+1/g(n)} \leq n^{1+o(1)}$. ◀

One property of note in the above proof is that uniformity can be applied to the circuits, without changing the argument. For example, if the small circuits for n^ε-circuit composition are LOGSPACE-uniform, then the small circuits for $\mathsf{TISP}[n^k, s(n)]$ are also LOGSPACE-uniform, assuming that k is constant (the argument becomes more complicated when adding an unbounded number of iterations).

▶ **Reminder of Theorem 2.** *For $0 < \varepsilon < 1$, the decision problem* General Circuit n^ε-Composition *does not have* LOGSPACE*-uniform $n^{1+o(1)}$ size circuits.*

Proof. Assume there is an $\varepsilon > 0$ such that General Circuit n^ε-Composition has LOGSPACE-uniform circuits of $n^{1+o(1)}$ size. Since Circuit n^ε-Composition is a special case of the general problem, it must also have such circuits. Therefore by Lemma 4, our assumption implies that for every constant $c \geq 1$, every problem in $\mathsf{TISP}[n^c, O(\log n)]$ has $n^{1+o(1)}$-size circuits as well. That is, we have

$$\mathsf{LOGSPACE} \subset \mathsf{SIZE}[n^{1+o(1)}]. \tag{1}$$

Since the circuits for General Circuit n^ε-Composition are LOGSPACE-uniform, there is an $O(\log n)$-space algorithm A that on input 1^n prints a $n^{1+o(1)}$-size circuit C_n computing General Circuit n^ε-Composition on n-bit instances. We are going to prove that General Circuit n^ε-Composition can be simulated in $n^{1+\varepsilon/2}$ time with about $n^{\varepsilon/2}$ bits of advice, implying a contradiction.

Similar to Santhanam and Williams [25], our next move is to define, for every rational $\alpha \in (0,1)$, a padded language $L_\alpha = \{(1^{n^\alpha}, n, i) \mid$ the ith bit of the circuit printed by $A(1^n)$ is $1\}$.

When $\alpha \in (0,1)$ is a fixed constant, we note the following properties of L_α:

(a) L_α is in LOGSPACE: on an m-bit instance $(1^{n^\alpha}, n, i)$, a machine deciding L_α only has to simulate A on 1^n in $O((\log m)/\alpha)$ space, and maintain a $(1 + o(1)) \log n$-bit counter for i, until the ith output bit of $A(1^n)$ is printed. (Note that $(1 + o(1)) \log n \leq O(\log m)$.) Hence by (1), L_α has an $n^{1+o(1)}$-size circuit family $\{D_m\}$, for every $\alpha > 0$.

(b) For an integer $n > 0$, if we want to know bits of the circuit printed by $A(1^n)$, the length of a relevant instance of L_α is only $|(1^{n^\alpha}, n, i)| \leq O(n^\alpha)$. Let $m(n)$ be the length of such an instance. On $m(n)$-bit instances, L_α outputs bits describing an $n^{1+o(1)}$-size circuit C_n which in turn solves n-bit instances of General Circuit n^ε-Composition.

Let $\alpha = \varepsilon/2$. We can decide General Circuit n^ε-Composition in $\tilde{O}(n^{1+\varepsilon/2})$ time with only $n^{\varepsilon/2+o(1)}$ bits of advice, as follows. On n-bit instances of the problem, our advice string is a description of the circuit $D_{m(n)}$ of size $m(n)^{1+o(1)}$ for $L_{\varepsilon/2}$ from item (a) above. From item (b), the length of the advice string is $m(n)^{1+o(1)} \leq n^{\varepsilon/2+o(1)}$. Given $D_{m(n)}$, our machine for circuit composition will evaluate $D_{m(n)}$ on $(1^{n^\alpha}, n, i)$ for all $i = 1, \ldots, n^{1+o(1)}$. This evaluation will, by definition, generate a description of an $n^{1+o(1)}$-size circuit C_n that solves n-bit instances of our problem. Sending the n-bit input to C_n, we can decide General Circuit n^ε-Composition in time $n^{1+o(1)} \cdot m(n)^{1+o(1)} \leq n^{1+\varepsilon/2+o(1)}$, with $n^{\varepsilon/2+o(1)}$ bits of advice.

However, since General Circuit n^ε-Composition is hard for $\mathsf{TIME}[n^{1+\varepsilon}]$ under $\tilde{O}(n)$-time reductions (Lemma 14), it follows from our simulation that *every* language in $\mathsf{TIME}[n^{1+\varepsilon}]$ is contained in $\mathsf{TIME}[n^{1+\varepsilon/2+o(1)}]/n^{\varepsilon/2+o(1)}$. This contradicts the time hierarchy theorem with sub-linear advice (which is folklore; see [25] for a proof). ◀

▶ **Reminder of Corollary 5.** *If n-IO Circuit n^ε-Composition has $\tilde{O}(n)$ size circuits, then*

$$\mathsf{TISP}\left[n^{(\log n)/\log\log n}, \tilde{O}(n)\right] \subseteq \mathsf{SIZE}[O(n^2)].$$

By a standard padding argument, we also have $\mathsf{TISP}\left[2^n, 2^{\sqrt{n \log n}}\right] \subseteq \mathsf{SIZE}\left[2^{O(\sqrt{n \log n})}\right]$.

Proof. We wish to maximize the time bound $n^{k(n)}$ in the consequence of Lemma 4 such that the hypothesis implies $O(n^2)$-size circuits for that time bound. This can be done by setting $n = (\log n)^{c_1 \cdot k(n)/\varepsilon}$ for a constant $c_1 > 0$. Solving for $k(n)$, we find $k(n) = c_2 \cdot \varepsilon(\log n)/(\log\log n)$ for a constant $c_2 > 0$. By Lemma 4, when $s(n)$-IO Circuit n^ε-Composition has $\tilde{O}(n)$ size circuits we have $\mathsf{TISP}[2^{c\varepsilon(\log n)^2/\log\log n}, O(n)] \subseteq \mathsf{SIZE}[n^2]$. By padding the input by $n \mapsto 2^{O(\sqrt{n \log n})}$, we also conclude that $\mathsf{TISP}[2^n, 2^{O(\sqrt{n \log n})}] \subseteq \mathsf{SIZE}[2^{O(\sqrt{n \log n})}]$. ◀

4 Log-Depth Circuit Lower Bounds

We now turn to proving uniform lower bounds for composing circuits of low depth. We can also prove an amplification lemma in this regime:

▶ **Lemma 16.** *Let $\varepsilon > 0$, $c, d \geq 1$, and $e \in [1, d)$. Let $k(n) = o(\log n/\log\log n)$ be constructible. If $(\log n)^d$-Depth Circuit n^ε-Composition has $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $n \cdot (\log n)^c$ size and $O((\log n)^e)$ depth, then every problem in $\mathsf{TISP}[n^{k(n)}, (\log n)^d]$ has $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $n \cdot (\log n)^{(c+1)\cdot k(n)/\varepsilon}$ size and $O((\log n)^e)$ depth.*

Proof. The proof is similar to Lemma 4. If we assume that $(\log n)^d$-Depth Circuit n^ε-Composition has $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $n \cdot (\log n)^c$ size and $O((\log n)^e$ depth, then we know that $(\log n)^d$-Depth Circuit n^ε-Composition can *compose its own circuit* in the same way that Circuit n^ε-Composition can, since the depth of the circuit is $(\log n)^e = o((\log n)^d)$.

As a result, if one step of a $\mathsf{SPACE}[s(n)]$ computation can be simulated with a $O(n)$-size $(\log n)^d$-depth circuit, then n^ε steps can be simulated with a $O(n \cdot (\log n)^{c+1})$ size circuit, $n^{2\varepsilon}$ steps can be simulated with a $O(n \cdot (\log n)^{2\cdot(c+1)})$ size circuit, and so on, using the $(\log n)^d$-Depth Circuit n^ε-Composition circuits. Since these are constructed by simply composing $(\log n)^d$-Depth Circuit n^ε-Composition circuits, if the original is $\mathsf{SPACE}[(\log n)^e]$-uniform and $O((\log n)^e)$ depth then all of the constructed circuits will be as well, as long as the circuit size remains polynomial-sized. By composing $k(n)/\varepsilon$ times,

we can construct $\mathsf{SPACE}[(\log n)^e]$-uniform $n \cdot (\log n)^{O(k(n)/\varepsilon)}$-size $O(\log n)^e$-depth circuits for $\mathsf{TISP}[n^{k(n)}, s(n)]$. Setting $s(n) = (\log n)^d$ completes the proof. ◀

▶ **Reminder of Theorem 3.** *For every $\varepsilon \in (0,1)$, $c \geq 1$, $d \in (1,2)$, and $e < d$, the problem $(\log n)^d$-Depth Circuit n^ε-Composition does not have $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $n \cdot (\log n)^c$ size and $O((\log n)^e)$ depth.*

Proof. Assume there are $\varepsilon > 0$, $d \in (1,2)$, and $e < d$ that $(\log n)^d$-Depth Circuit n^ε-Composition has $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $n \cdot (\log n)^c$ size and $O((\log n)^e)$ depth. Setting $k(n) := (\log n)^{d-1}$, we have $d - 1 < 1$ by assumption, therefore $k(n) = o(\log n / \log\log n)$. Applying Lemma 16 to our assumption, we can infer that the class $\mathsf{SPACE}[(\log n)^d] = \mathsf{TISP}[n^{O((\log n)^{d-1})}, (\log n)^d]$ has $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $n \cdot (\log n)^c$ size and $O((\log n)^e)$ depth. That is, every problem in $\mathsf{SPACE}[(\log n)^d]$ also has $\mathsf{SPACE}[(\log n)^e]$-uniform circuits of $O((\log n)^e)$ depth. But by Proposition 12,

$$\mathsf{SPACE}[(\log n)^e]\text{-uniform } \mathsf{DEPTH}[(\log n)^e] \subseteq \mathsf{SPACE}[(\log n)^e],$$

so we have actually derived $\mathsf{SPACE}[(\log n)^d] \subseteq \mathsf{SPACE}[(\log n)^e]$. This contradicts the space hierarchy theorem [27], because $e < d$. ◀

4.1 Depth Lower Bound for SAT

Now we show that the SAT problem does not have subquadratic-space-uniform $\tilde{O}(n)$-size circuits of $\log^{2-\varepsilon} n$ depth, for all $\varepsilon > 0$. The results here will take the typical form of "alternation-trading proofs" [31], where one quickly simulates a space-bounded computation with alternations, removes the alternations using efficient (assumed) SAT circuits, and attempts to prove a contradiction. The key difference between our proof approach and prior ones is that we are able to use the space hierarchy theorem to establish the contradiction, which leads to a stronger space and depth lower bound.

We will use the following powerful simulation lemma of Reischuk (based on Nepomnjascii [20]) in our proof.

▶ **Lemma 17** ([23], p.282). *For constructible functions $t_1(n)$, $t_2(n)$, $s(n)$ and $a(n)$, we have the containment*

$$\Sigma_{a(n)}\mathsf{TISP}[t_1(n)^{t_2(n)}, s(n)] \subseteq \Sigma_{a(n)+t_2(n)}\mathsf{TISP}[a(n) \cdot s(n) + t_1(n) \cdot t_2(n) \cdot s(n), t_1(n) \cdot s(n)].$$

In general, Reischuk's lemma shows how alternating computations with large time bounds and small space bounds can be converted into alternating computations with much lower time bounds and slightly larger space bounds.

We need another lemma showing how good SAT circuits can yield circuits for alternating computations. It is similar to other arguments of this kind, but we include it for completeness:

▶ **Lemma 18.** *Let $c, d > 0$. If $SAT \in \mathsf{SPACE}[(\log n)^d]$-uniform $\mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^c, (\log n)^d]$, then for $k \geq 1$*

$$\Sigma_k\mathsf{TIME}[n] \subseteq \mathsf{SPACE}[(\log n)^d]\text{-uniform } \mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^{(c+1)\cdot k}, (\log n)^d].$$

Proof. The proof is by induction on k, the number of alternations. For $k = 1$, the statement becomes $\mathsf{NTIME}[n] \subseteq \mathsf{SPACE}[(\log n)^d]$-uniform $\mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^{c+1}, (\log n)^d]$, which is true by assumption.

Suppose that for some fixed value of k the lemma holds. Consider some $L \in \Sigma_{k+1}\mathsf{TIME}[n]$. We can then construct a circuit of size $n \cdot (\log n)^{(c+1)\cdot(k+1)}$ and depth $O((\log n)^d)$ using $O((\log n)^d)$ space that computes L. Since L is verified in linear time, the number of bits of nondeterminism in the first alternation is at most linear, so there is a $\Pi_k\mathsf{TIME}[n]$ verifier $V(x, y)$ for L that takes both the input x to L and the first set of nondeterministic bits y as input and checks whether y is a valid witness that $x \in L$. By assumption, V has $\mathsf{SPACE}[(\log n)^d]$-uniform $\mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^{(c+1)\cdot k}, (\log n)^d$ circuits.

Consider the Circuit-SAT instance that consists of the above circuit that computes V as well as the input x that is meant to be the input of the original language L. The description of this circuit is of size $N = n \cdot (\log n)^{(c+1)\cdot k+1}$, so by assumption there is a circuit of size $N \cdot (\log N)^c = n \cdot (\log n)^{(c+1)\cdot k+1} \cdot (O(\log n))^c = n \cdot (\log n)^{(c+1)\cdot(k+1)}$ and depth $O((\log N)^d) = O((\log n)^d)$ computable in $\mathsf{SPACE}[(\log N)^d] = \mathsf{SPACE}[(\log n)^d]$ that solves this SAT instance.

Both the Π_k verifier circuit for L and the Circuit-SAT circuit can be constructed in $\mathsf{SPACE}[(\log n)^d]$, and by hard-coding the description of the verifier circuit as input to the Circuit-SAT instance, the resulting circuit will solve L. Furthermore the size and depth of the circuit is simply the size and depth of the Circuit-SAT instance, since the Π_k circuit is fed as a description (only the size of the description matters). This circuit then exists for every $L \in \Sigma_{k+1}\mathsf{TIME}[n]$. Therefore if

$$\Sigma_k\mathsf{TIME}[n] \subseteq \mathsf{SPACE}[(\log n)^d]\text{-uniform } \mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^{(c+1)\cdot k}, (\log n)^d]$$

then

$$\Sigma_{k+1}\mathsf{TIME}[n] \subseteq \mathsf{SPACE}[(\log n)^d]\text{-uniform } \mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^{(c+1)\cdot(k+1)}, (\log n)^d]$$

and the induction holds for all $k \geq 1$.

Note that the above argument holds even if the circuits are non-uniform. Furthermore, as long as the size of the circuits remain fairly small, k can also be a function of the input size. If $k(n) = o(\log n/\log\log n)$, then the circuits produced will still have $n^{1+o(1)}$-length descriptions at every step of the induction, since

$$n \cdot (\log n)^{(c+1)\cdot o(\log n/\log\log n)} \leq n \cdot n^{o(c+1)} = n^{1+o(1)}.$$

The main factor that determines the size, depth and uniformity of the next alternation in the induction is the size of the previous circuit in the induction. Therefore, as long as that circuit has $\tilde{O}(n)$ size, the inductive step will hold. ◀

▶ **Reminder of Theorem 6.** *For all $d < 2$, and all $c \geq 1$,*

$$SAT \notin \mathsf{SPACE}[\log^d n]\text{-}uniform\ \mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^c, (\log n)^d]\,.$$

Proof. Assume that SAT $\in \mathsf{SPACE}[(\log n)^d]$-uniform $\mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^c, (\log n)^d]$. We will show that this assumption contradicts the space hierarchy theorem.

By Lemma 18, we conclude for constructible $k(n) = o(\log n/\log\log n)$ that

$$\Sigma_{k(n)}\mathsf{TIME}[n] \subseteq \mathsf{SPACE}[(\log n)^d]\text{-uniform } \mathsf{SIZE\text{-}DEPTH}[n \cdot (\log n)^{(c+1)\cdot k(n)}, (\log n)^d]. \quad (2)$$

Set $s(n) = (\log n)^{d'}$ in the following, where $d < d' < 2$. Applying Lemma 17 with $a(n) = 0$, $t_1(n) = O(n)$, $t_2(n) = s(n)/\log n$, and $s(n)$ arbitrary, we have the containment

$$\mathsf{SPACE}[s(n)] = \mathsf{TISP}[2^{O(s(n))}, s(n)] \subseteq \Sigma_{s(n)/\log n}\mathsf{TIME}[n \cdot s(n)^2/\log n].$$

Setting $k(n) = s(n)/\log n$ and noting that $s(n)/\log n = (\log n)^{d'-1} = o(\log n/\log\log n)$, we derive from the containment (2) that

$$\Sigma_{s(n)/\log n}\mathsf{TIME}[n \cdot s(n)^2/\log n]$$

is contained in

$$\begin{aligned}
&\mathsf{SPACE}[(\log n)^d]\text{-uniform } \mathsf{SIZE\text{-}DEPTH}[n \cdot s(n)^2 \cdot (\log n)^{(c+1)\cdot s(n)/\log n}, O((\log n)^d)] \\
&\subseteq \mathsf{SPACE}[(\log n)^d]\text{-uniform } \mathsf{DEPTH}[O((\log n)^d)] \\
&\subseteq \mathsf{SPACE}[o(s(n))]\text{-uniform } \mathsf{DEPTH}[o(s(n))] \\
&\subseteq \mathsf{SPACE}[o(s(n))]. \qquad \text{(Proposition 12)}
\end{aligned}$$

We have derived $\mathsf{SPACE}[s(n)] \subseteq \mathsf{SPACE}[o(s(n))]$, which contradicts the space hierarchy theorem [27]. ◀

5 Conclusion

In this paper, we showed how to "amplify" small-circuit upper bounds in a new way: if a simple circuit composition problem has nearly-linear size circuits, then a much larger class of problems also has nearly-linear size circuits. This led to new circuit lower bounds and connections between lower bound problems. Many open problems have naturally arisen.

- We have shown that $\mathsf{TIME}[n^{1+\varepsilon}]$ does not have $\mathsf{LOGSPACE}$-uniform linear-size circuits, and the lower bound is non-relativizing. Can this be strengthened to P-uniform linear circuits? Alternatively, can our lower bounds for circuit composition be generalized to prove that $\mathsf{TIME}[n^k] \not\subseteq \mathsf{LOGSPACE}$-uniform $\mathsf{SIZE}[n^{k-\varepsilon}]$, for any constant k?
  We conjecture that the circuit composition problems defined in this paper, especially General Circuit n-Composition, require *non-uniform* super-linear-size circuits. The fact that we can at least rule out $\mathsf{LOGSPACE}$-uniform circuits gives some hope that future work can relax the uniformity conditions.
- What additional consequences can be derived from assuming $\mathsf{NP} \subset \mathsf{SIZE}[O(n)]$? How well can PSPACE-complete problems like QBF be solved with circuits, under this assumption? From the results of this paper, we have that QBF has $2^{\tilde{O}(\sqrt{n})}$-size circuits, assuming SAT is in $\mathsf{SIZE}[O(n)]$ or assuming $\mathsf{TIME}[n^{1+\varepsilon}]$ is in $\mathsf{SIZE}[O(n)]$. Is it possible that $\mathsf{NP} \subset \mathsf{SIZE}[O(n)]$ implies $\mathsf{PSPACE} \subset \mathsf{P/poly}$?
- Can we prove $\mathsf{P} \not\subset \mathsf{P}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}(O(n))$? Is the problem equivalent to $\mathsf{P} \not\subset \mathsf{SIZE}(O(n))$? A yes-answer would show that constructing these linear-size circuits cannot even be done by a P^{NP} process, progressing even closer to $\mathsf{P} \not\subset \mathsf{SIZE}(O(n))$. We observe that $\mathsf{P} \not\subset \mathsf{SIZE}(O(n))$ is in fact equivalent to $\mathsf{P} \not\subset \mathsf{P}^{\Sigma_2\mathsf{P}}$-uniform $\mathsf{SIZE}(O(n))$: in $\mathsf{P}^{\Sigma_2\mathsf{P}}$ one can guess and verify a linear-size circuit for a polynomial-time computation.

Acknowledgments. We thank the CCC referees for their helpful comments.

References

1 Miklos Ajtai. Determinism versus non-determinism for linear time RAMs (extended abstract). In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1999.

2 Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

3 Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 295–302, 2001.

4 Sanjeev Arora and Boaz Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, Cambridge, 2009.

5 Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003.

6 Samuel R. Buss and Ryan Williams. Limits on alternation trading proofs for time-space lower bounds. *Computational Complexity*, 24(3):533–600, 2015.

7 Stephen A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *STOC*, pages 338–345, 1979.

8 Ning Ding. Some new consequences of the hypothesis that P has fixed polynomial-size circuits. In *Theory and Applications of Models of Computation TAMC*, pages 75–86, 2015.

9 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:166, 2015. URL: `http://eccc.hpi-web.de/report/2015/166`.

10 Lance Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60(2):337–353, April 2000.

11 Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):833–865, 2005.

12 Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed polynomial size circuit bounds. In *Proceedings of 24th Annual IEEE Conference on Computational Complexity*, pages 19–26, 2009.

13 Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285–306, 1965.

14 Frederick Hennie and Richard Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

15 John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, April 1977.

16 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 62(4):512–530, 2001.

17 Richard Lipton. Some consequences of our failure to prove non-linear lower bounds on explicit functions. In *Proceedings of 9th Annual Structure in Complexity Theory Conference*, pages 79–87, 1994.

18 Richard J. Lipton and Anastasios Viglas. Non-uniform depth of polynomial time and space simulations. In *Proceedings of Fundamentals of Computation Theory, 14th International Symposium (FCT)*, pages 311–320, 2003.

19 Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity*, 22(2):311–343, 2013.

20 V. Nepomnjascii. Rudimentary predicates and turing calculations. *Soviet Mathematics – Doklady*, 11(6):1462–1465, 1970.

21 Nicholas Pippenger. On simultaneous resource bounds (preliminary version). In *FOCS*, pages 307–311, 1979.

22 Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

23 Rüdiger Reischuk. *Einführung in die Komplexitätstheorie*. Teubner, 1990.

24 Walter L. Ruzzo, Janos Simon, and Martin Tompa. Space-bounded hierarchies and probabilistic computations. *J. Comput. Syst. Sci.*, 28(2):216–230, 1984.

25 Rahul Santhanam and Ryan Williams. On uniformity and circuit lower bounds. *Computational Complexity*, 23(2):177–205, 2014. Preliminary version in CCC'13.

26 Richard Stearns, Juris Hartmanis, and Philip Lewis. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190. IEEE, 1965.

27 Richard Stearns, Juris Hartmanis, and Philip Lewis. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190. IEEE, 1965.

28 Dieter Van Melkebeek. *A survey of lower bounds for satisfiability and related problems*, volume 7. Now Publishers Inc, 2007.

29 Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 1999.

30 R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Computational Complexity*, 17(2):179–219, 2008.

31 Ryan Williams. Alternation-trading proofs, linear programming, and lower bounds. *TOCT*, 5(2):6, 2013.

32 Christopher Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31(2):169–181, 1985.

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

A Oracle Relative to Which Polytime has Small Constructible Circuits

In the following, we use the standard model of oracle computation for logarithmic space of Ruzzo, Simon, and Tompa [24]: there is an special oracle tape that is write-only, its content does not count towards the space bound, and the oracle tape is erased after each oracle query.

▶ **Theorem 19.** *For every $k \geq 1$, there is an oracle B such that every language solvable in time n^k with an oracle for B has $O(n)$-size B-oracle circuits constructible in logspace equipped with an oracle for B.*

Proof. Our construction is very similar to that of Wilson [32], who shows there are oracles relative to which P has size $O(n)$ circuits. To make the circuits logspace-constructible as well, we add a simple but crucial modification to the oracle, which relies on having a fixed polynomial upper bound on the running time.

Fix a constant $k \geq 1$. In the following, let $\{M_i\}$ be an enumeration of machines running in at most $n^k + k$ steps, and let $\langle \cdot, \cdot \rangle$ be an efficient pairing function. We construct the oracle B in stages. In stage 0, we assign B to be the empty set.

In stage n:

- Start with an empty set S_n. For every integer $1 \leq i \leq n$ and n-bit string x, execute M_i on x with the current oracle B for $n^k + k$ steps. If M_i accepts x, put the pair $\langle i, x \rangle$ in S_n.
- "Mark" every string that is queried by the M_i's among these 2^n executions. Note there are at most $(n^k + k)n2^n$ strings marked in this step, and the total number of marked strings (over all stages $0, \ldots, n$) at this point is at most $(n^k + k)n2^{n+1}$.
- Let y_n be a string of length $n + (k+1)\log n + 3k$ such that $\{\langle i, x \rangle y_n \mid \langle i, x \rangle \in S_n\}$ contains no marked strings. There are $t = n^{k+1}2^{n+3k}$ such strings y_n, so we are considering $t > (n^k + k)n2^{n+1}$ different sets of strings over $\{0,1\}^{2n+(k+1)\log n+3k}$. Hence at least one of the sets does not contain a marked string. Put all $\langle i, x \rangle y_n$ in the oracle B.

- Finally, put the set of strings $\{0^{n^{k+1}+k+1}1j \mid \text{the } j\text{th bit of } y_n \text{ is } 1\}$ in B as well, where the j's are construed as $\lceil \log_2(n + k \log n + 3k + 1) \rceil$-bit strings. Note that none of the strings in this set can be marked in stage n, because all of them have length greater than $n^{k+1} + (k+1)$, and no M_i can query a string longer than $n^k + k$ over inputs x of length n.

Now, for any machine M_i and sufficiently large $n \gg i$, our circuit C_n computing M_i on all inputs of length n consists of the single gate

$$B(\langle i, x \rangle y_n),$$

with the index i and the string y_n hard-coded in C_n. By our construction of B, there is a chosen string y_n of length $O(n)$ for which deciding $\langle i, x \rangle y_n \in B$ is equivalent to deciding if M_i accepts x. The total number of wires (and hence size) in the circuit C_n is $O(n)$.

Furthermore, for every machine M_i running in n^k time, the circuits C_n for M_i can also be constructed in $O(k \log n)$ space with an oracle for B. Given the string 1^n, our logspace machine L_i for printing C_n first prints the index i of M_i in some natural encoding, and prints n "sources" of C_n, indicating the inputs $x_1, \ldots, x_n$. To construct the string y_n, L_i uses the oracle B. In particular, the logspace machine L_i has a counter which holds some integer $j = 1, \ldots, n + k \log n + 3k$. For each j, L_i prints $0^{n^{k+1}+k+1}1j$ on the write-only oracle tape (by maintaining a counter $\ell = 1, \ldots, n^{k+1} + k + 1$ for printing the zeroes) then queries B. The j-th query answer tells L_i the j-th bit of y_n, so L_i can output these bits as part of the description of C_n. ◀

This oracle is especially interesting when contrasted with the lower bound of Santhanam and Williams [25] that $\mathsf{P} \not\subset \mathsf{P}$-uniform $\mathsf{SIZE}(O(n))$. The proof of that lower bound *does* relativize (and thus is true for all oracles). The key difference between $\mathsf{P} \not\subset \mathsf{P}$-uniform $\mathsf{SIZE}(O(n))$ and the (non-relativizing) lower bound of Theorem 2 (general circuit composition is not in LOGSPACE-uniform $n^{1+o(1)}$ size) seems to be that in the former, there is no fixed polynomial upper bound on the complexity class (P) that is being simulated.

B Subexponential-Size Circuits for QBF from Small-Size Circuits for SAT

We can also derive interesting new consequences from the assumption that the SAT problem has $\tilde{O}(n)$-size circuits. They follow without much difficulty from the literature on SAT time-space tradeoffs, but we feel the connections are worth recording.

▶ **Claim 20** ([10]). *If SAT* $\in \mathsf{TIME}[O(n)]$ *then there is a* $c > 0$ *such that for all* k, $\Sigma_k\mathsf{TIME}[O(n)] \subseteq \mathsf{TIME}[n \cdot (\log n)^{ck}]$.

Proof. It is enough to show that SAT $\in \mathsf{TIME}[O(n)]$ implies that

$$\Sigma_k\mathsf{TIME}[O(n)] \subseteq \Sigma_{k-1}\mathsf{TIME}[n \cdot (\log n)^d]$$

for a fixed constant $d > 0$. Suppose SAT $\in \mathsf{TIME}[O(n)]$. Then by an efficient Cook-Levin Theorem (see for example [11]), we have $\mathsf{NTIME}[O(n)] \subseteq \mathsf{TIME}[\tilde{O}(n)]$ and $\mathsf{coNTIME}[O(n)] \subseteq \mathsf{TIME}[\tilde{O}(n)]$. Let $L \in \Sigma_k\mathsf{TIME}[O(n)]$ have an acceptance condition of the form:

$$\exists x_1 \forall x_2 \ldots Q_k x_k M(x, x_1, x_2, \ldots, x_k) \tag{3}$$

where M is a deterministic $O(n)$-time machine, and all x_i's are $O(n)$-length strings. Then on the tuple of strings $(x, x_1, x_2, \ldots, x_{k-1})$, the expression

$$Q_k x_k M(x, x_1, x_2, \ldots, x_k) \tag{4}$$

represents a computation in $\mathsf{coNTIME}[O(n)]$, which by hypothesis is computable in $\tilde{O}(n)$ time. Let N be a deterministic machine that computes the value of (4) in $\tilde{O}(n)$ time, given $(x, x_1, x_2, \ldots, x_{k-1})$ as input. Then deciding the truth of

$$\exists x_1 \forall x_2 \ldots Q_{k-1} x_{k-1} N(x, x_1, \ldots, x_{k-1})$$

is equivalent to deciding (3). Thus L can be decided in $\Sigma_{k-1}\mathsf{TIME}[\tilde{O}(n)]$, which completes the proof. ◀

▶ **Corollary 21.** *If* $SAT \in \mathsf{TIME}[O(n)]$ *then* $\mathsf{LOGSPACE} \subseteq \bigcup_k \mathsf{TIME}[n \cdot (\log n)^k]$.

Proof. We know that for every $L \in \mathsf{LOGSPACE}$ there is a k such that $L \in \Sigma_k\mathsf{TIME}[O(n)]$. Apply the above claim. ◀

▶ **Claim 22.** *If* $SAT \in \mathsf{SIZE}[O(n)]$ *then there is a* c *such that for all* k, $\Sigma_k\mathsf{TIME}[O(n)] \in \mathsf{SIZE}[n(\log n)^{ck}]$.

Proof. Similar to Claim 20. If $\mathrm{SAT} \in \mathsf{SIZE}[O(n)]$ then $\mathsf{NTIME}[O(n)] \subseteq \mathsf{SIZE}[\tilde{O}(n)]$, which means that

$$\Sigma_k\mathsf{TIME}[O(n)] \subseteq \Sigma_{k-1}\mathsf{TIME}[n \log^c n]/(n \log^c n) \subseteq \ldots$$

$$\subseteq \mathsf{TIME}[n \log^{ck} n]/(n \log^{ck} n) \subseteq \mathsf{SIZE}[n(\log n)^{ck}]\,. \quad ◀$$

▶ **Corollary 23.** *If* $SAT \in \mathsf{SIZE}[O(n)]$ *then* $\mathsf{LOGSPACE} \subseteq \bigcup_k \mathsf{SIZE}[n \cdot (\log n)^k]$.

Specifically, we can relate the circuit complexity of SAT and QBF as follows:

▶ **Reminder of Theorem 7.** *If* SAT *is in* $\mathsf{SIZE}[\tilde{O}(n)]$ *then* QBF *is in* $\mathsf{SIZE}[2^{O(\sqrt{n \log n})}]$.

Proof. Suppose $\mathrm{SAT} \in \mathsf{SIZE}[\tilde{O}(n)]$. By Claim 22, we have

$$\Sigma_{k(n)}\mathsf{TIME}[O(n)] \subseteq \mathsf{SIZE}[n(\log n)^{c \cdot k(n)}]. \tag{5}$$

Let $a(n) = 0$, $s(n) = O((\log n)^2/\log\log n)$, $t_2(n) = k(n)$, and $t_1(n) = n$. Applying Lemma 17, we have

$$\mathsf{TISP}[n^{k(n)}, O((\log n)^2/\log\log n)] \subseteq \Sigma_{k(n)}\mathsf{TIME}[\tilde{O}(n)]. \tag{6}$$

Setting $k(n) = \varepsilon \log n/(\log\log n)$ for sufficiently small $\varepsilon > 0$, we have

$$\mathsf{SPACE}[\varepsilon(\log n)^2/\log\log n] \subseteq \mathsf{TISP}[n \cdot 2^{\varepsilon(\log n)^2/\log\log n}, \varepsilon(\log n)^2/\log\log n]$$

$$\subseteq \mathsf{TISP}[n^{1+\varepsilon \log n/\log\log n}, \varepsilon(\log n)^2/\log\log n] \subseteq \Sigma_{1+\varepsilon \log n/\log\log n)}\mathsf{TIME}[\tilde{O}(n)]$$

$$\subset \mathsf{SIZE}[n(\log n)^{1+\varepsilon \log n/\log\log n)}] = \mathsf{SIZE}[n^{1+2\varepsilon}].$$

By padding each language in $\mathsf{SPACE}[\varepsilon(\log n)^2/\log\log n]$ to size $N = 2^{O(\sqrt{n \log n})}$, we have $(\log N)^2/\log\log N \leq O(n \log n/\log n) \leq O(n)$ and $N^2 \leq 2^{O(\sqrt{n \log n})}$, we can conclude that $\mathsf{SPACE}[O(n)]$, which includes QBF, has circuits of size $2^{O(\sqrt{n \log n})}$. ◀