The Circuit-Input Game, Natural Proofs, and Testing Circuits With Data*

Brynmor Chapman†
Stanford University
chapmanb@cs.stanford.edu

Ryan Williams‡
Stanford University
rrw@cs.stanford.edu

December 10, 2014

Abstract

We revisit a natural zero-sum game from several prior works. A *circuit player*, armed with a collection of Boolean circuits, wants to compute a function f with one (or some) of its circuits. An *input player* has a collection of inputs, and wants to find one (or some) inputs on which the circuit player cannot compute f. Several results are known on the existence of small-support strategies for zero-sum games, in particular the above circuit-input game. We give two new applications of these classical results to circuit complexity:

Natural properties useful against self-checking circuits are equivalent to circuit lower bounds. We show how the Natural Proofs barrier may be potentially sidestepped, by simply focusing on analyzing *circuits that check their answers*. Slightly more precisely, we prove $\mathsf{NP} \not\subset \mathsf{P/poly}$ *if and only if* there are natural properties that (a) accept the SAT function and (b) are useful against polynomial-size circuits that never err when they report SAT. (Note, via self-reducibility, any small circuit can be turned into one of this kind!) The proof is very general; similar equivalences hold for other lower bound problems. Our message is that one should search for lower bound methods that are designed to succeed (only) against circuits with "one-sided error."

Circuit Complexity versus Testing Circuits With Data. We reconsider the problem of program testing, which we formalize as deciding if a given circuit computes a (fixed) function f. We define the "data complexity" of f (as a function of circuit size s) to be the minimum cardinality of a *test suite* of inputs: a set of input/output pairs necessary and sufficient for deciding if any given circuit of size at most s computes a slice of f. (This is a "gray-box testing" problem, where the value s is side information.) We prove that designing small test suites for f is equivalent to proving circuit lower bounds on f: the data complexity of testing f is "small" if and only if the circuit complexity of f is "large." Therefore, circuit lower bounds may be constructively viewed as *data design* circuit-testing problems.

1 Introduction

We consider the following zero-sum game studied in prior work (e.g., [LY94, FIKU08]), which we call the *circuit-input game*. Fix a Boolean function f. A *circuit player* chooses from a set of Boolean circuits, while an *input player* chooses from a set of inputs. A payoff goes to the circuit player if its chosen circuit computes f on the chosen input; otherwise, the input player is paid.

*A preliminary version of this work appears in ITCS'15.

†Supported by NSF CCF-1212372. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

‡Supported by NSF CCF-1212372.

To our knowledge, the circuit-input game was first explicitly studied by Lipton and Young [LY94], in the context of providing complexity-theoretic applications of strategies for general zero-sum games (see also [Yao77]). Among other results, they (and independently, Newman [New91] and Althöfer [Alt94]) proved that approximate and succinct strategies exist for any zero-sum game in the sense for an $m \times n$ matrix M, there exists a strategy for the row player with support size $O(\log n)$, and a strategy for the column player with support size $O(\log m)$, which together additively approximate the optimal strategy of the game. This result was used to prove the existence of so-called *anti-checkers*: for any function $f : \{0,1\}^n \to \{0,1\}$ with circuit complexity at least s, there exists a set S of $O(s)$ n-bit inputs on which all circuits of size at most s/n fail to compute f correctly on a $1/2 - \varepsilon$ fraction of inputs in S.

In this paper, we start by proving a general extension of these classical results, and we also prove that our extension cannot be improved in a certain technical way. We then use our extension to present two new applications of these classical results to the field of circuit complexity.

Natural Properties Equivalent to Lower Bounds In the first part of the paper, we give an alternative view into the Natural Proofs barrier [RR97]; in particular, we suggest a new pathway around it. Recall that Natural Proofs have three properties.

- They are *constructive*: they contain an efficient algorithm A from a complexity class Γ for testing Boolean functions given as truth tables,
- They are *large*: algorithm A accepts a large fraction (at least $1/2^{O(n)}$) of all n-bit Boolean functions, and
- They are *useful*: algorithm A rejects all functions which are truth tables of circuits from a circuit class $\mathscr{C}$, for infinitely many input lengths.

Such an algorithm A is called a Γ-*natural property useful against* $\mathscr{C}$, and Razborov-Rudich showed that there are no P/poly-natural properties useful against P/poly unless every pseudorandom function candidate can be broken. Hence a natural proof is not capable of proving P/poly lower bounds, or statements like $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$. This ruled out many potential methods for proving circuit lower bounds.

We show that a minor (and in hindsight, obvious) modification to the "useful" condition of Natural Proofs not only makes the barrier disappear, but it makes circuit lower bounds equivalent to the existence of such modified natural properties. Our minor modification is perhaps best illustrated by considering NP vs P/poly and the SAT problem (although any self-reducible NP-complete problem would suffice). We begin with the well-known observation that any polynomial-size circuit C can be assumed, without loss of generality, to never err when it reports satisfiability. That is, given a circuit C that potentially solves SAT, C can be augmented to print a satisfying assignment: create a larger circuit C' that contains copies of C, such that when C reports "SAT" on a formula, C' repeatedly plugs in values for variables of the formula and queries C on the reduced formulas to check if satisfiability still holds. Either C will eventually be in error (in which case C' reports "UNSAT") or C will produce a SAT assignment, in which case C' reports "SAT" without error.

Let C be a Boolean circuit on n inputs. Define C to be a *SAT solver* if for all n-bit formulas F, $C(F) = 1$ implies that F is satisfiable. (We call such circuits "SAT solvers" because one could use these circuits to print satisfying assignments to formulas, when the circuits report "SAT.") The class of functions computable by polynomial-size SAT solvers is an expressive class, including special cases such as 2-SAT, Horn-SAT, etc. By the previous paragraph, to prove $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$ it suffices to prove that no polynomial-size family of SAT solvers can compute SAT. That is, a lower bound proof only needs to be *useful* against small SAT solvers. But *how* might we prove this? If we used combinatorial or probabilistic methods, we might expect to find an efficient test for Boolean functions, such that random functions (and SAT) pass,

but SAT solver circuits do not pass. This looks very much like Natural Proofs, except instead of rejecting all polynomial-size circuits, our test will only try to reject the polynomial-size SAT solvers. It turns out that such tests always exist, if $\mathsf{NP} \not\subset \mathsf{P/poly}$. More formally, let SAT_n denote the restriction of SAT to formulas encoded in n bits. We prove:

Theorem 1.1 $\mathsf{NP} \not\subset \mathsf{P/poly}$ *if and only if there is an* $\mathsf{AC}^0/n^{o(1)}$*-natural property that is useful against polynomial-size SAT solvers and accepts* SAT_n *for all n.*

Compare with the main theorem of Razborov-Rudich: if there are $\mathsf{P/poly}$-natural properties useful against $\mathsf{P/poly}$, then there are no strong pseudorandom generators. The above theorem suggests that, to circumvent "naturalness" and prove circuit lower bounds for SAT, we should try to look for proof methods which *fail* on arbitrary polynomial-size circuits, but *succeed* on circuits that try to print full satisfying assignments. This point also gives intuition for why Theorem 1.1 holds without hurting cryptography: the truth tables of SAT solvers do not look at all like random functions, so a natural property useful against SAT solvers is in no danger of distinguishing pseudorandom functions from truly random ones.[1]

Equivalences similar to Theorem 1.1 hold for other circuit lower bound problems. In general, for any function f that permits a zero-error or one-sided error corrector in a complexity class $\mathscr{C}$, there is an equivalence between proving $f \notin \mathscr{C}$ and exhibiting natural properties useful against "error-corrected f-solving circuits." We also show how a version of our statement holds for f with (randomized) polynomial-time program checkers [BK95].

Testing Circuits for Functionality Using Data In the second part of the paper, we apply succinct strategies for zero-sum games to set up a framework that is "dual" to the usual computational view of circuits computing functions on inputs, treating inputs as the "programs" and circuits as the "input data". We generalize Lipton-Young's anti-checkers, showing how the general problem of circuit lower bounds can be seen in a constructive light: as designing small data sets that can be used to conclusively test whether a given circuit computes a particular function. This also yields a complexity-theoretic perspective on the practical problem of software testing (see [Pat05, UL07]).

To describe the results, we need to fix some terminology. Let $\mathscr{C}$ be the collection of Boolean circuits over the standard basis B_2, the set of all two-bit Boolean functions. The *size* of a circuit $C \in \mathscr{C}$ is measured by its number of gates; let $|C|$ denote the size of C. Let $f : \{0,1\}^\star \to \{0,1\}$ be a decision problem. The nth slice of f is the function $f_n : \{0,1\}^n \to \{0,1\}$ that agrees with f on all n-bit inputs. Function f_n is computed by $C_n \in \mathscr{C}$ if for every $x \in \{0,1\}^n$ we have $C_n(x) = f_n(x)$, and the *circuit complexity of* f is the function $g : \mathbb{N} \to \mathbb{N}$ such that $g(n)$ equals the minimum $|C_n|$ over all $C_n \in \mathscr{C}$ computing f_n.

Define the computational problem of TESTING FOR f, which we abbreviate as TEST-f, to be the union (over all n) of the set of *all* circuits C_n which compute f_n. For simplicity, we define the *sth slice of* TEST-f to be the set of all size-s circuits computing some slice of f (rather than the set of all circuits encoded in s bits that compute some slice).

Let $\{0,1\}^{\leq n}$ be the set of all bit strings of length at most n. Let $X_s \subseteq \{0,1\}^{\leq s} \times \{0,1\}$. That is, X_s is a collection of pairs of $(\leq s)$-bit strings with bit labels.

Definition 1.1 *The set* X_s *is said to* test size-s for f *if:*

- *for all* $(x,b) \in X_s$, $f(x) = b$, *and*
- *for every size-s circuit* $C \in (\mathscr{C} \setminus \text{TEST-}f)$ *with n inputs (that is,* $C \neq f_n$*), there is a pair* $(x,b) \in X_s$ *such that* $|x| = n$ *and* $C(x) \neq b$.[2]

[1] As a reviewer succinctly noted, one might interpret this result as saying, "pseudorandom functions have no hope of computing SAT, so why should we care about them?"

[2] Note that s is the maximal number of input gates in a circuit of size s, so $n \leq s$.

That is, a circuit C of size s and n inputs agrees with f on all n-bit inputs in X_s if and only if the circuit C computes f_n. Hence such a collection X_s can be used to conclusively test whether a given circuit computes the relevant slice of f.

An infinite family of input sets $\{X_s \mid s \in \mathbb{N}^+\}$ is said to *test for* f if for all $s \geq 1$, and for every circuit $C \in (\mathscr{C} \setminus \text{TEST-}f)$ of size s, there is a string $x \in X_s$ such that $C(x) \neq f_{|x|}(s)$. Thinking of TEST-f as a decision problem to be computationally solved, the family $\{X_s\}$ can serve as a *test suite* for deciding this problem, by providing pairs $(x, f(x)) \in X_s$, then verifying that a given circuit C agrees with all pairs.

Using inputs to test for whether a circuit computes f leads to a natural definition of the complexity of testing for f:

Definition 1.2 *The* data complexity of TEST-f *is defined to be the function* $g : \mathbb{N} \to \mathbb{N}$ *such that* $g(s)$ *equals the minimum cardinality of a set* X_s *testing size-*s *for* f.

(Note that we measure data complexity as a function of the circuit size s, because s measures the "length of the input" to the problem TEST-f.) The design of a test suite $\{X_s\}$ with low complexity is an extremely natural goal that is rather well-motivated practically. It is a *data design* problem, as opposed to the typical algorithm/circuit design problem. Just as there are trivial circuits of $O(n2^n)$ size for every Boolean function on n variables, there are also trivial test suites where each X_s contains $2^{O(s)}$ inputs of length up to s. The theory of circuits becomes interesting when we restrict the complexities of circuits; the theory of test suites becomes similarly interesting when restricting the amount of necessary data.

There is an inherent duality between data complexity and circuit complexity. Our main theorem is that lower bounds on the circuit complexity of f are basically *equivalent* to upper bounds on the data complexity of testing f. This also uses the circuit-input game in a crucial way:

Theorem 1.2 *Let* $f : \{0,1\}^\star \to \{0,1\}$, *and let* $S(n) \geq 2n$ *for all* n.

1. *If* f *is in* $\mathsf{SIZE}(S(n))$, *then the data complexity of testing size-*s *circuits for* f *is at least* $2^{\Omega(S^{-1}(s))}$ *almost everywhere.*
2. *If* f *is not in* $\mathsf{SIZE}(n \cdot S(n))$, *then the data complexity of testing size-*s *circuits for* f *is at most* $O(2^{S^{-1}(s)} + S^{-1}(s) \cdot s^2 \log s)$ *infinitely often.*

As a corollary, we can give a different characterization of the $\mathsf{NP} \not\subset \mathsf{P/poly}$ problem: it is equivalent to the existence of a test suite for testing circuits for SAT with subexponential data complexity:

Corollary 1.1 $\mathsf{NP} \not\subset \mathsf{P/poly}$ *(resp.* $\mathsf{NP} \not\subset \mathsf{i.o.P/poly}$*) if and only if for every* $\varepsilon > 0$ *and for infinitely many* s *(resp. for every* $\varepsilon > 0$ *and for every* s*), the data complexity of testing size-*s *circuits for SAT is at most* $O\left(2^{s^\varepsilon}\right)$.

We strongly recommend reading (not skipping) the Preliminaries section below, as it introduces notation, background, and intuitions that will be required at key points in the paper.

2 Preliminaries

2.1 Background

Complexity Theory We assume standard background in complexity theory, along with the following additional notation: $\mathsf{SIZE}(S(n))$ is the class of functions $f : \{0,1\}^\star \to \{0,1\}$ such that f is computable with a size-$S(n)$ circuit family $\{C_n\}$ over the basis B_2 of all two-bit Boolean functions.

Zero-Sum Games and Succinct Strategies We think of a zero-sum game as an $m \times n$ matrix M with entries from $[0,1]$ describing the payoff to the row player in a game between a *row player* and *column player*. We now describe the concepts and theorems of Lipton and Young [LY94] relevant to this paper.

Definition 2.1 *Let S be any set, and let $k \in \mathbb{N}$. A* k-uniform distribution on S *is a probability distribution obtained by choosing uniformly from a multiset of k elements from S.*

Definition 2.2 *Let $\mathscr{C}$ be a set of n circuits, let $\mathscr{I}$ be a set of m inputs, and let M be an $m \times n$ matrix with entries in $[0,1]$. (Intuitively, $M(C,x)$ represents some cost of the computation of $C(x)$.) The* circuit-input game w.r.t. *M is the two-player zero-sum game given by the matrix M.*

For a function $f : \{0,1\}^\star \to \{0,1\}$, our central game of interest is what we call the *circuit-input game for f on size-s circuits and n inputs*, which is the circuit-input game w.r.t. the following matrix M. M has 2^n rows (for all strings x in $\{0,1\}^n$) and $2^{O(s \log s)}$ columns (for all circuits C of size s), and

$$M(C,x) := \begin{cases} 0 & \text{if } C(x) = f(x) \\ 1 & \text{otherwise} \end{cases}$$

Let $\mathscr{C}$, $\mathscr{I}$, and M be as above. Assume without loss of generality that $\exists C_0, x_0$ such that $M(C_0,x_0) = 0$ and $\exists C_1, x_1$ such that $M(C_1,x_1) = 1$. We shall use the following two theorems saying that approximately optimal strategies with small support exist for every game M:

Theorem 2.1 ([New91, Alt94, LY94]) *Let $\varepsilon > 0$, let $k > \dfrac{\ln|\mathscr{I}|}{2\varepsilon^2}$, and let $\ell > \dfrac{\ln|\mathscr{C}|}{2\varepsilon^2}$.*

1. *There exists a k-uniform distribution p on $\mathscr{C}$ such that for every $x \in \mathscr{I}$, the expectation $\mathop{\mathbf{E}}_{C \sim p}[M_{C,x}] < \mathscr{V}(M) + \varepsilon$, where $\mathscr{V}(M)$ denotes the value of the circuit-input game w.r.t. M.*
2. *There exists an ℓ-uniform distribution p on $\mathscr{I}$ such that for every $C \in \mathscr{C}$, the expectation $\mathop{\mathbf{E}}_{x \sim p}[M_{C,x}] > \mathscr{V}(M) - \varepsilon$, where $\mathscr{V}(M)$ denotes the value of the circuit-input game w.r.t. M.*

This theorem can be proved using a random sampling argument and standard large deviation (Chernoff-Hoeffding) bounds. From Theorem 2.1, we may derive the following general consequence which does not appear in prior work:

Theorem 2.2 *Let $\mathscr{C}_n$ be a set of 2^t circuits where each circuit has n inputs, let $\mathscr{I}_n \subseteq \{0,1\}^n$, and let $f : \{0,1\}^n \to \{0,1\}$. Let $\varepsilon : \mathbb{N} \to \mathbb{Q} \cap (0,1]$, and let $p, q : \mathbb{N} \to [0,1]$ with $p + q \leq 1 - \varepsilon$. For every $n \in \mathbb{N}$, one of the following must hold:*

1. *There exists an $O(n/\varepsilon(n)^2)$-size multiset $X_n \subseteq \mathscr{C}_n$ such that for every $y \in \mathscr{I}_n$, $C(y) = f(y)$ for more than a $p(n)$ fraction of the circuits $C \in X_n$.*
2. *There exists an $O(t/\varepsilon(n)^2)$-size multiset $Y_n \subseteq \mathscr{I}_n$ such that for every $C \in \mathscr{C}_n$, $C(y) \neq f(y)$ for more than a $q(n)$ fraction of the inputs $y \in Y_n$.*

Proof. Let M be the circuit-input game for a function f. Let ε, p, q, and n be as in the statement of the theorem. Set a parameter $\delta := \varepsilon(n)/2$, set $k := \dfrac{\ln|\mathscr{I}_n|}{\delta^2} = \dfrac{O(n)}{\delta^2}$, and set $\ell := \dfrac{\ln|\mathscr{C}_n|}{\delta^2} = \dfrac{O(t)}{\delta^2}$. Then by Theorem 2.1, there exists a k-uniform distribution X_n on $\mathscr{C}_n$ such that for all $y \in \mathscr{I}_n$,

$$\mathop{\mathbf{E}}_{C \sim X_n}[M[C,y]] < \mathscr{V}(M) + \delta, \tag{1}$$

and there exists an ℓ-uniform distribution Y_n on $\mathscr{I}_n$ such that for every $C \in \mathscr{C}_n$,

$$\mathop{\mathbf{E}}_{y \sim Y_n}[M[C,y]] > \mathscr{V}(M) - \delta. \tag{2}$$

Assume that there exists $y^* \in \mathscr{I}_n$ such that $C(y^*) \neq f(y^*)$ for at least a $1 - p(n)$ fraction of the circuits $C \in X_n$. Since M is a Boolean matrix, we have for every $C^* \in \mathscr{C}_n$ that

$$\begin{aligned} q(n) &\leq 1 - p(n) - \varepsilon(n) \\ &\leq \mathop{\mathbf{Pr}}_{C \sim X_n}[C(y^*) \neq f(y^*)] - \varepsilon(n) \quad \text{(by choice of } y^*\text{)} \\ &< \mathscr{V}(M) - \delta \quad \text{(by choice of } \delta \text{ and (1))} \\ &< \mathop{\mathbf{Pr}}_{y \sim Y_n}[C^*(y) \neq f(y)]. \quad \text{(by (2))} \end{aligned}$$

This completes the proof. □

That is, we can trade off between the "measure of success" p of the succinct strategy for the circuit player, and the measure of success q of the succinct strategy for the row player. This tradeoff can be exploited for complexity-theoretic purposes as follows: if item 1 does not hold (because of circuit lower bounds for computing f with $\mathscr{C}$ circuits) then there are small multisets "witnessing" this inability to compute. Lipton and Young observed this consequence for the special case of $p = 1/2$ and $q = 1/2 - \varepsilon$, calling such sets *anti-checkers*. In our main results, we shall adjust p and q to different values, as needed. (For example, in our results concerning natural properties, we use the case of $p = 0$ and $q = 1 - \varepsilon$.)

It is natural to ask whether Theorem 2.2 can be improved, so that $p + q = 1$. We now show that this is not possible, at least not in full generality. In particular, while Theorem 2.2 holds for all $p + q \leq 1 - \varepsilon$, where $\varepsilon > 0$, it does not hold for $p = q = 1/2$ and all matrices M:

Theorem 2.3 *Theorem 2.2 does not hold with $p = q = 1/2$.*

Proof. Let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function without circuits of size $c \cdot ns$, for a sufficiently large constant $c \geq 1$. Suppose Theorem 2.2 holds with $p = q = 1/2$ and the circuit-input game for f on size-s circuits and n inputs. Then at least one of the following holds.

1. There is an $O(n)$-size multiset $\mathscr{C}_n$ of size-s circuits where $\Pr_{C \in \mathscr{C}_n}[C(x) \neq f(x)] < 1/2$ holds for all $x \in \{0,1\}^n$.
2. There exists an $O(s \log s)$-size multiset $\mathscr{X}_n$ of n-bit inputs such that for every circuit C of size s, $\Pr_{x \in \mathscr{X}_n}[C(x) \neq f(x)] \geq 1/2$.

In the first case, f can be implemented with a (strict) MAJORITY circuit on $\mathscr{C}_n$, giving a circuit of size $O(ns)$, contradicting our choice of f.

The second case also leads to a contradiction. For any $\mathscr{X}_n$, we may take a circuit C of size $O(n)$ that has a single input x in $\mathscr{X}_n$ hardwired along with the value $f(x)$. The circuit C outputs $f(x)$ on input x, and otherwise it outputs the bit b maximizing the quantity $\Pr_{x' \in \mathscr{X}_n \setminus \{x\}}[f(x') = b]$. It follows that $\Pr[C(x) \neq f(x)] < \frac{1}{2}$.

□

2.2 Additional Prior Work

The problem of approximately solving a zero-sum game has been studied in operations research as well; for example, Grigoriadis and Khachiyan [GK95b] show how to find an approximately optimal strategy with randomness in time sublinear in the size of the game matrix.

Bshouty *et al* [BCG+96] studied the circuit-input game in the context of learning theory, focusing on the complexity of finding succinct strategies in the game. They proved (for example) that if $\mathsf{NP} \subset \mathsf{P/poly}$ then one can uniformly construct circuits solving SAT in $\mathsf{ZPP}^{\mathsf{NP}}$ – this gave a new collapse of the polynomial-hierarchy under the assumption that $\mathsf{NP} \subset \mathsf{P/poly}$. Dually, if $\mathsf{NP} \not\subset \mathsf{P/poly}$, then their results also show that one can uniformly construct multisets of satisfiable formulas that "fool" all small circuits, in $\mathsf{ZPP}^{\mathsf{NP}}$: Fortnow, Pavan, and Sengupta applied this consequence to the "two queries" problem in structural complexity [FPS08]. Subsequently, Fortnow *et al* [FIKU08] proved that the problem of finding approximate succinct strategies in an implicitly represented game (such as the circuit-input game) is promise-S_2-complete. Other works on succinct games include [CR08, SV12].

When circuit lower bounds are true, succinct strategy results like Theorem 2.2 tell us that there are small distributions of inputs that "fool" all small circuits. Another class of related results have focused on a different flavor of hardness result: *given the code* of an efficient algorithm (randomized or otherwise) that's supposed to solve SAT, one can construct even smaller distributions that fool the given algorithm [GSTS05, Ats06, BTW10].

Our ideas for using data to test circuits for a function are vaguely related to the notion of *teaching dimension of a class of concepts*, from learning theory (see Goldman and Kearns [GK95a] and Shinohara and Miyano [SM91]). The teaching dimension is defined with respect to a *collection* of functions, and it bounds the total number of labeled examples needed to identify each concept in the collection (to distinguish it from the other concepts). However, this notion is information-theoretic: there are no computational bounds placed on *what* is distinguishing one function from another. In our setting, we have a collection of *programs* and a specific function f of interest, and wish to know how many labeled examples we need to distinguish "bad" programs which do not compute f from "good" ones which do.

Mulmuley's GCT program [Mul11] has also considered "sets of counterexamples" similar to our test sets, calling them "obstructions."

3 Natural Properties Equivalent to Circuit Lower Bounds

In this section, we prove that natural properties useful against self-checking circuits are *equivalent* to circuit lower bounds in some important settings.

Let SAT_n denote the restriction of SAT to formulas encoded in n bits.

Reminder of Theorem 1.1 $\mathsf{NP} \not\subset \mathsf{P/poly}$ *if and only if there is an* $\mathsf{AC}^0/n^{o(1)}$*-natural property that is useful against polynomial-size SAT solvers and accepts* SAT_n *for all n.*

Proof. One direction of the equivalence is trivial: if there is any logical property that is false on the truth tables of all polynomial-size SAT_n solvers for infinitely many n, yet the property is true of SAT_n for all n, then no polynomial-size SAT solving circuit can compute SAT_n almost everywhere. Therefore $\mathsf{NP} \not\subset \mathsf{P/poly}$.

Now we proceed with the other direction. Assume $\mathsf{NP} \not\subset \mathsf{P/poly}$. Let s be a polynomial in n. For every $n \in \mathbb{N}$, let $\mathscr{C}_n$ denote the set of n-input circuits of size $s(n)$ which are SAT solvers (i.e., they never err on unsatisfiable formulas). Set $\mathscr{I}_n := \mathrm{SAT}_n$, i.e., the set of satisfiable formulas encoded in n bits. Consider the circuit-input game M for SAT, over the set of circuits $\mathscr{C}_n$ and set of inputs $\mathscr{I}_n$.

Applying Theorem 2.2 to this game M (taking $p(n) = 0$ and $q(n) = 1 - \varepsilon(n)$ for some inverse polynomial $\varepsilon(n)$), either:

1. there is an $\mathrm{poly}(s,n)$-size set $X \subseteq \mathscr{C}_n$ such that for every $x \in \mathscr{I}$, at least one $C \in X$ computes $\mathrm{SAT}(x)$, or

2. there is an $\mathrm{poly}(s,n)$-size set $Y \subseteq \mathscr{I}_n$ such that every circuit $C \in \mathscr{C}_n$ computes SAT correctly on at most an $\varepsilon(n)$ fraction of inputs in Y.

In the first case, we may construct a polynomial-size circuit for SAT_n by simply taking the OR of the circuits in X: since the circuits never err on unsatisfiable formulas, this will compute SAT_n. Our assumption $\mathsf{NP} \not\subset \mathsf{P/poly}$ is therefore contradicted if the first case holds for almost every n.

Suppose the second case holds for infinitely many n. Then we can construct an algorithm A which takes as input the 2^n-bit truth table of a function $f : \{0,1\}^n \to \{0,1\}$ and is armed with Y as advice. Algorithm A accepts f if f outputs 1 on at least a $2\varepsilon(n)$ fraction of the inputs in Y, and rejects f if f outputs 1 on at most an $\varepsilon(n)$ fraction of the inputs in Y. This A is implementable in polynomial-size AC^0 (in fact, depth-3 AND-OR-AND circuits), by classical results on distinguishing strings with many 1's from strings with many 0's [Ajt83, Vio11]. Furthermore, A trivially accepts SAT_n for every n, because $\mathrm{SAT}_n(y) = 1$ for every input $y \in Y$.

Notice that, while A rejects the truth tables of SAT solving circuits of $s(n)$ size, A will *accept* a randomly chosen Boolean function with probability $1 - o(1)$. Notice that the advice needed is $s(n) \leq O(n^k)$, which is polylogarithmic in the input length, 2^n. Therefore, A is an $\mathsf{AC}^0/n^{o(1)}$-natural property that is useful against SAT solving circuits of $s(n)$ size. Such an A can be constructed for every polynomial $s(n)$, assuming $\mathsf{NP} \not\subset \mathsf{P/poly}$. □

A reviewer pointed out that the above theorem holds not only for SAT solvers, but for any circuit which computes a NP-complete problem with one-sided error, regardless of whether the problem in question exhibits self-reducibility. We consider SAT specifically because self-reducibility allows us to construct a SAT solver from an arbitrary circuit with only polynomial overhead.

3.1 Natural Properties from Circuit Lower Bounds for Checkable Functions

It is easy to extend the previous theorem to general functions f with deterministic one-sided checkers: functions f that allow polynomial-size circuits with oracle gates for f which never err when they output 0 (or never err when they output 1). Now we consider languages which have (randomized) polynomial time program checkers, such as EXP-complete sets [BFL91, BK95], the Permanent [Lip91], etc. Such randomized program checkers can be adapted to give polynomial size circuit families which act as program checkers that *deterministically* check all functions computable by small circuits. These deterministic program checkers can then be used to prove (as above with SAT) equivalences between circuit lower bounds and the existence of natural properties. Here we use the following definition of a program checker.

Definition 3.1 *Let $L \in \{0,1\}^*$. A* randomized (polynomial time) program checker for L *is a randomized (polynomial time) algorithm with oracle access to an arbitrary function f, which when given an n-bit input x and randomness r will accept with probability greater than $2/3$ (over the choice of r) if $L = f$ and will reject with probability greater than $2/3$ if $L(x) \neq f(x)$.*

In order for a randomized program checker to be adapted to produce a deterministic circuit family, we require that the language L be paddable, so that a circuit computing L on inputs of length n can also compute L on inputs of length less than n. Hence for the rest of the section, we consider only functions which are paddable in the following sense.

Definition 3.2 *A language L is* paddable *if there exists a language L' such that $L = \left\{x01^k : x \in L', k \in \mathbb{N}\right\}$.*

Note that any language L' can be converted into a paddable language L as above, while preserving both asymptotic circuit complexity (up to a polynomial factor) and the existence of program checkers.

Theorem 3.1 *Let p and t be polynomials, and let L be a paddable language with a randomized $t(n) - n$ time program checker. Then there exists a polynomial size circuit family which deterministically checks all functions on n inputs computable by circuits of size at most $p(t(n))$.*

Proof. Let p and t be polynomials in n. Let L be any paddable language with a randomized $t(n)-n$ time program checker, i.e. there exists a randomized $t(n)-n$ time algorithm A such that for every input $x \in \{0,1\}^n$ and every $f : \{0,1\}^{t(n)} \to \{0,1\}$, $A^f(x) = 1$ with probability more than $2/3$ if f is the $t(n)$th slice of L, and $A^f(x) = 0$ with probability more than $2/3$ if $f\left(x1^{t(n)-n}\right) \neq L(x)$.

We may amplify this success probability to $1-2^{-q(n)}$ (for some polynomial $q(n) > 2p(t(n))\log p(t(n)) + n$), by creating another checker A' which runs A independently $18q(n)$ times and returns the MAJORITY of the $18q(n)$ results; appealing to a Chernoff bound yields the higher success probability.

We may simulate A' with a polynomial size family $\{B_n\}$ of oracle circuits which take as input the string $x \in \{0,1\}^n$ and a string of randomness $r \in \{0,1\}^{n^k}$, and which use oracle gates to compute the function f. Since there are at most $2^{2p(t(n))\log p(t(n))}$ circuits of size $p(t(n))$ and 2^n n-bit input strings, we have (using a union bound) with non-zero probability (over the choice of $r \in \{0,1\}^{n^k}$), for every $x \in \{0,1\}^n$ and every f computable by a circuit of size $p(t(n))$,

- $f(z) = L(z)$ for all $z \in \{0,1\}^{t(n)} \implies B_n^f(x,r) = 1$ and
- $f(x) \neq L(x) \implies B_n^f(x,r) = 0$.

Hence there is some choice of randomness $r^* \in \{0,1\}^{n^k}$ for which $B_n(-,r^*)$ is a deterministic program checker (where the randomness r^* is hard coded into the circuit). □

Fix a polynomial p and a circuit family A_n which deterministically checks circuits of size at most $p(n)$. Now for any circuit $\{C\}$ with $t(n)$ inputs and of size $p(t(n))$, we may augment C with $A_{t(n)}$ to create a self-checking circuit C' on n inputs with polynomial overhead. We may treat C' as a ternary circuit which outputs 0 (resp. 1) if $A_{t(n)}$ outputs 1 and C outputs 0 (resp. 1), and which outputs $\perp$ if $A_{t(n)}$ outputs 0. We may treat $\perp$ as 0 or 1, in which case C' gives a circuit with one-sided error (in either direction). Call all such C' the *self-checked circuits for L*.

Theorem 3.2 *Let L be a paddable language with a randomized polynomial time program checker. If $L \notin \mathsf{P}/poly$, then for every polynomial s, there exists a natural property computable in $\mathsf{AC}^0/n^{o(1)}$ useful against size-$s(n)$ self-checked circuits for L.*

Proof. In the case where $\perp = 0$ above, we may take $\mathscr{C}_n$ to be the set of $s(n)$ size circuits, $\mathscr{I}_n := \{0,1\}^n \cap L$, and $M(C,y) := 1 - C(y)$. From Theorem 2.2 (taking $p(n) = 0$ and $q(n) = 1-\varepsilon(n)$ for some inverse polynomial ε), either there is a polynomial size set $X \subseteq \mathscr{C}_n$ such that for every $y \in \mathscr{I}$, at least one $C \in X$ computes $L(y)$, or there is a polynomial size set $Y \subseteq \mathscr{I}_n$ such that every circuit $C \in \mathscr{C}_n$ computes L correctly on at most an $\varepsilon(n)$ fraction of inputs in Y. If the former case holds for almost every n, then we may construct a polynomial size circuit family for L by taking the OR of the circuits in X. Otherwise, the latter case holds infinitely often, giving an efficiently computable ($\mathsf{AC}^0/n^{o(1)}$) natural property useful against functions with one-sided error which are computable with $s(n)$ size circuits.

Note that the choice to treat $\perp$ as 0 is arbitrary. We may instead treat $\perp$ as 1, in which case Theorem 2.2 either gives a polynomial size set of circuits whose AND computes L, or an $\mathsf{AC}^0/n^{o(1)}$-computable natural property useful against functions with $p(n)$ size circuits and which are (bitwise) at least L. □

4 Circuit Lower Bounds as Data Design Problems

We now turn to our results on testing circuits and data complexity. In the following, let $f : \{0,1\}^\star \to \{0,1\}$ and let $S : \mathbb{N} \to \mathbb{N}$ satisfy $S(n) \geq n$ for all n. For simplicity, we prove the main theorem (Theorem 1.2) only for the class of circuits over the basis B_2 of all two-bit Boolean functions, although analogous statements will hold for any complete basis with minor modifications. For a circuit C, let $n(C)$ be its number of inputs.

Recall that the *data complexity* of testing size-s circuits for f is the minimum cardinality of a set S of labeled examples $(x, f(x))$ (where $|x|$ can range from 1 to s) that suffice to distinguish all size-s circuits which do not compute a slice of f from those which do. Namely, for all size-s circuits C which do not compute $f_{n(C)}$, there is some "witness" in S: a pair $(x, f(x)) \in S$ with $|x| = n(C)$ such that $C(x) \neq f(x)$.

The data complexity of testing size-s circuits is always at most $2^{O(s)}$ for any function f: one can simply include all possible input/output pairs on inputs of length up to s. We are interested to know: for what functions f can the data complexity be much smaller? We prove an equivalence between upper bounds on the data complexity of testing f and lower bounds on the circuit complexity of f:

Reminder of Theorem 1.2 *Let $f : \{0,1\}^\star \to \{0,1\}$, and let $S(n) \geq 2n$ for all n.*

1. *If f is in $\mathsf{SIZE}(S(n))$, then the data complexity of testing size-s circuits for f is at least $2^{\Omega(S^{-1}(s))}$ almost everywhere.*
2. *If f is not in $\mathsf{SIZE}(n \cdot S(n))$, then the data complexity of testing size-s circuits for f is at most $O(2^{S^{-1}(s)} + S^{-1}(s) \cdot s^2 \log s)$ infinitely often.*

Since for a uniformly random function $f_n : \{0,1\}^n \to \{0,1\}$, $f_n \notin \mathsf{SIZE}(2^n/n^2)$ with high probability, we have the following corollary:

Corollary 4.1 *If $f : \{0,1\}^* \to \{0,1\}$ is uniformly random, then almost certainly the data complexity of testing size-s circuits for f is at most $O(s^2 \log^2 s)$ infinitely often.*

We also note that since the circuit complexities of a function f over any complete bases differ by at most a constant factor (and as noted previously, all results presented hold for an arbitrary complete gate basis with minor modifications to the proofs), the data complexities of TEST-f over any complete bases differ by at most a constant factor.

To get some intuition towards a proof of Theorem 1.2, notice that if we replace "data complexity" with "time complexity" in the above, one direction of the equivalence is easy to establish. Namely, for functions f computable within exponential time, when the circuit complexity of f is large, the *time complexity* of testing circuits for f will be provably low, as follows.

Suppose $S(n)$ is a lower bound on the circuit complexity of computing f on n-bit inputs. To efficiently test a given circuit C of size s with n inputs, we can immediately reject if $s < S(n)$, otherwise we may try all $2^n < 2^{S^{-1}(s)}$ inputs to C and check whether the truth table obtained for C matches f_n on n-bit inputs. For f computable within $2^{O(n)}$ time, this algorithm takes $2^{O(S^{-1}(s))}$ time; larger $S(n)$ entails a faster running time. (For example, if some f in $2^{O(n)}$ time requires $2^{\varepsilon n}$ size $\mathscr{C}$-circuits, then testing $\mathscr{C}$ for f is in $\mathrm{poly}(s)$ time.) However, this particular connection is not terribly useful: we are basically saying that strong circuit lower bounds happen to make testing circuits for f trivial, because most circuits can be immediately rejected. Moreover we do not know if low time complexity for testing circuits for f will imply analogous circuit lower bounds for f, in general.

The equivalence between circuit complexity and data complexity is far less obvious. We use the following consequence of results on the circuit-input game (Theorem 2.2 in the Preliminaries): when circuits are too small to compute a function, there are small data sets that will efficiently refute these small circuits.

Proof of Theorem 1.2. (Part 1.) Suppose for all n, there is a circuit C of size $S(n)$ which computes f_n on all inputs of length n. For each n, we claim that every test set T_s for f on circuits of size $S'(n) = S(n) + n$ satisfies $|T_{s'}| \geq 2^n$. Observe that for every circuit C of size $S(n)$ with n inputs and for all x of length n, there is a circuit C_x of size at most $S'(n) = S(n) + n$ which agrees with C on all inputs except x, where C_x and C disagree. In particular, C_x can use a tree of $n-1$ gates that outputs 1 if and only if the input equals x, and take the XOR of this tree's output with the circuit C.

So given an n-input circuit C computing f with size at most $S'(n)$, in order to distinguish C from all of the C_x, x must be included in $T_{s'}$. That is, all x of length $n = \Omega(S^{-1}(s))$ must be in $T_{s'}$.

(Part 2.) Suppose the circuit complexity of f is greater than $n \cdot S(n)$ on inputs of length n. Then there cannot be a collection $\mathscr{D}$ of $O(n/\varepsilon^2)$ size-$S(n)$ circuits such that for all $x \in \{0,1\}^n$, $C(x) = f(x)$ for more than a $1/2$ fraction of C in $\mathscr{D}$: otherwise, taking the MAJORITY of the outputs of circuits in $\mathscr{D}$ would yield a circuit for f of complexity at most $n \cdot S(n)$. Therefore, item 1 of Theorem 2.2 does not hold with $p = 1/2$, and hence item 2 must hold for $q \leq 1/2 - \varepsilon$ for every sufficiently small ε.

Setting ε appropriately, this implies for all input lengths m ranging from n to $n \cdot S(n)$, there is a set $X_{m,s}$ of m-bit strings x with labels $f(x)$ and cardinality $O(S(n)\log S(n))$, such that every circuit of size $S(n)$ taking m bits of input fails to compute f correctly on at least $1/10$ of the x in the set $X_{m,s}$. By adding all strings in $X_{m,s}$ to the set X_s, we can refute any circuit with more than n inputs and size $S(n)$, with a set of $O(nS(n)^2 \log S(n))$ strings.

For input lengths m that are below n, there may be a size $S(n)$ circuit for f that works on all m-bit strings. To cover this case, we simply include (with labels for the value of f) all bit strings of length up to $n-1$ in the set X_s as well – then, every circuit of size s and at most n inputs can also be checked. In total, we have a test set X_s of cardinality $O(2^n + nS(n)^2 \log S(n))$. For size s circuits, we set $s := S(n)$, i.e., $n = S^{-1}(s)$, so the cardinality is $O(2^{S^{-1}(s)} + S^{-1}(s) \cdot s^2 \log s)$ as a function of the circuit size s. □

In the above theorems, the small cardinality test suites X_s have the following structure: for input sizes which are "too long" to support size-s circuits for f, we have small sets of counterexamples from the circuit-input game, but as the input sizes decrease, we reach a threshold where it's possible to compute f within size s, and must start including all possible inputs and their labels.

When we consider polynomial-size circuits in general, we simply obtain an *equivalence*:

Corollary 4.2 *A function f is in P/poly if and only if for some $\varepsilon > 0$, the data complexity of testing circuits for f is greater than 2^{s^ε} for almost every s.*

Proof. If f is in P/poly, then it is in $\mathsf{SIZE}(n^k)$ for some constant k. By Part 1 of Theorem 1.2, the data complexity of testing circuits for f is at least $2^{\Omega(s^{1/k})}$. On the other hand, if f is not in P/poly, then it is not in $\mathsf{SIZE}(n^{k+1})$ for every k. By Part 2 of Theorem 1.2, the data complexity of testing circuits for f is then at most $O(s^k \log s + 2^{s^{1/k}})$ for all k. □

Reminder of Corollary 1.1 $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$ *(resp.* $\mathsf{NP} \not\subset \mathsf{i.o.P}/\mathrm{poly}$*) if and only if for every $\varepsilon > 0$ and for infinitely many s (resp. for every $\varepsilon > 0$ and for every s), the data complexity of testing size-s circuits for SAT is at most $O\left(2^{s^\varepsilon}\right)$.*

5 Conclusion

There are many questions raised by this work that seem worth exploring further, regarding the circumvention of natural proofs and regarding the testing of circuits for computing functions. Here are three questions we particularly like.

1. *Can new circuit lower bounds be proved, based on the guidance of Theorem 1.1?* Again, this theorem tells us that we should expect there to be combinatorial properties useful against polynomial-size SAT solving circuits, or in general, circuits which never err when they print solutions to their input instances.

   To be more concrete, let $\mathrm{CLIQUE}^{n/2}_{n^2} : \{0,1\}^{n^2} \to \{0,1\}^n$ be the function which treats its input as an $n \times n$ adjacency matrix A, and outputs a bit vector specifying a clique of size at least $n/2$ in A, when

one exists (otherwise, it outputs the all-zeros vector). Can one prove that computing $\text{CLIQUE}_{n^2}^{n/2}$ requires circuits of size at least $4n^2$ over the basis of all fan-in two Boolean functions?

2. *Can the equivalence of Theorem 1.2 be tightened further?* Currently there is a gap between the two implications in the equivalence, amounting to a multiplicative factor of n. Could this gap be necessary?

3. *How does the complexity of f relate to the complexity of testing circuits for f?* Here we mean "complexity" in the usual, most generic sense: if f is known to be computable in some particular complexity class, what can we say about the complexity class(es) that support testing for f?

Acknowledgements We thank the ITCS reviewers for their helpful comments. In particular, one reviewer improved an earlier version of Theorem 2.3.

References

[Ajt83] Miklos Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[Alt94] Ingo Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199:339–355, 1994.

[Ats06] Albert Atserias. Distinguishing SAT from polynomial-size circuits, through black-box queries. In *IEEE Conference on Computational Complexity*, pages 88–95. IEEE, 2006.

[BCG+96] Nader Bshouty, Richard Cleve, Ricard Gavalda, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(2):268–286, 1996.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42:269–291, 1995.

[BTW10] Andrej Bogdanov, Kunal Talwar, and Andrew Wan. Hard instances for satisfiability and quasi-one-way functions. In *ICS*, pages 290–300, 2010.

[CR08] Venkatesan T. Chakaravarthy and Sambuddha Roy. Finding Irrefutable Certificates for S_2^p via Arthur and Merlin. In *25th International Symposium on Theoretical Aspects of Computer Science*, volume 1 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 157–168, Dagstuhl, Germany, 2008.

[FIKU08] Lance Fortnow, Russell Impagliazzo, Valentine Kabanets, and Christopher Umans. On the complexity of succinct zero-sum games. *Computational Complexity*, 17(3):353–376, 2008.

[FPS08] Lance Fortnow, Aduri Pavan, and Samik Sengupta. Proving SAT does not have small circuits with an application to the two queries problem. *J. Comput. Syst. Sci.*, 74(3):358–363, 2008.

[GK95a] Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. *J. Comput. Syst. Sci.*, 50(1):20–31, 1995.

[GK95b] M.D. Grigoriadis and L.G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995.

[GSTS05] 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. See also CCC'05.

[Lip91] Richard Lipton. New directions in testing. In *Distributed Computing and Cryptography, vol. 2 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. American Mathematical Society, 1991.

[LY94] Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *STOC*, pages 734–740. ACM, 1994.

[Mul11] Ketan D. Mulmuley. On P vs. NP and geometric complexity theory: Dedicated to Sri Ramakrishna. *J. ACM*, 58(2):5, 2011.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39:67–71, 1991.

[Pat05] Ron Patton. *Software Testing (2nd Edition)*. Sams, Indianapolis, IN, USA, 2005.

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

[SM91] Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Comput.*, 8(4):337–347, 1991.

[SV12] Grant Schoenebeck and Salil P. Vadhan. The computational complexity of nash equilibria in concisely represented games. *TOCT*, 4(2):4, 2012.

[UL07] Mark Utting and Bruno Legeard. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[Vio11] Emanuele Viola. Randomness buys depth for approximate counting. In *FOCS*, pages 230–239. IEEE, 2011.

[Yao77] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS*, pages 222–227. IEEE, 1977.