Proving SAT does not have Small Circuits with an Application to the Two Queries Problem

Lance Fortnow * A. Pavan † Samik Sengupta ‡

Abstract

We show that if SAT does not have small circuits, then there must exist a small number of satisfiable formulas such that every small circuit fails to compute satisfiability correctly on at least one of these formulas. We use this result to show that if $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, then the polynomial-time hierarchy collapses to $\mathrm{S}_2^p \subseteq \Sigma_2^p \cap \Pi_2^p$. Even showing that the hierarchy collapsed to Σ_2^p remained open prior to this paper.

1 Introduction

Bshouty, Cleve, Gavaldá, Kannan and Tamon [BC$^+$96] give a probabilistic algorithm with a SAT oracle that learns circuits given hypothesis and membership queries to that circuit. If SAT has polynomial-size circuits, then one can use their algorithm to give a probabilistic procedure, once again with a SAT oracle, that finds that circuit. One can verify in co-NP that this circuit correctly computes SAT.

What if SAT does not have small circuits? Can one find a short witness of this fact? We give an affirmative answer. Building on Bshouty et al. we show that if SAT does not have polynomial-size circuits at length n, then for every k there are

*Department of Computer Science, University of Chicago, Chicago, IL 60637. Email: fortnow@cs.uchicago.edu

†Department of Computer Science, Iowa State University, Ames, IA 50011. Part of the work done while the author was a postdoc at NEC Research Institute. Research supported in part by NSF grant CCF-0430807. Email: pavan@cs.iastate.edu

‡Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260. Email: samik@cse.buffalo.edu

polynomial number of satisfiable formulas such that every circuit of size at most n^k fails to give a correct answer on at least one of these formulas.

In addition, one can find these formulas with a probabilistic algorithm with a SAT oracle. These satisfiable formulas along with satisfying assignments give a co-NP verifiable proof that SAT does not have n^k-size circuits.

We show an application to the following well studied question: Is one query to SAT as powerful as two queries to SAT? In the context of computing functions, Krentel [Kre88] showed that if any function that can be computed by two queries to SAT can be computed by one query, then $\mathrm{P} = \mathrm{NP}$, i.e, if $\mathrm{PF}^{\mathrm{NP}[1]} = \mathrm{PF}^{\mathrm{NP}[2]}$, then $\mathrm{P} = \mathrm{NP}$. It is natural to ask whether we can obtain such collapse if we focus on languages instead of functions.

Kadin [Kad88] showed that if $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, then the polynomial-time hierarchy collapses to Σ_3^p. Wagner [Wag87] showed that the collapse can be improved to $\Delta_3^p = \mathrm{P}^{\Sigma_2^p}$. Beigel, Chang, and Ogihara [BCO93], building on the work of Wagner [Wag87] and Chang and Kadin [CK96] obtained a stronger conclusion. They showed that every language in the polynomial-time hierarchy can be solved by a polynomial-time machine that makes at most one NP query and one Σ_2^p query.

Buhrman and Fortnow [BF99] showed many other collapses including that polynomial-time hierarchy collapses to $\mathrm{BPP}^{\mathrm{NP}}$. They tried to improve their collapse to Σ_2^p but they could not find a way to easily determine whether SAT had small circuits.

Using our lemma we can solve this problem and achieve the collapse. We show that if $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$ then PH collapses to $\mathrm{S}_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}([\text{Cai01}]) \subseteq \Sigma_2^p \cap \Pi_2^p$.

2 Preliminaries

Given $k > 0$, $\mathrm{P}^{\mathrm{NP}[k]}$ denotes the class of languages accepted by a polynomial-time-bounded oracle Turing machine that makes at most k adaptive queries to SAT.

The class S_2^p has been defined independently by Russell and Sundaram [RS98] and Canetti [Can96]. A set L is in S_2^p if there is a polynomial-time predicate R and a polynomial $p(\cdot)$ such that

$$\begin{aligned} x \in L &\Rightarrow \exists y\, \forall z\, R(x,y,z), \text{ and} \\ x \notin L &\Rightarrow \exists z\, \forall y\, \neg R(x,y,z), \end{aligned}$$

where $|y|, |z| \leq p(|x|)$.

The class S_2^p can be viewed as a game among two competing provers and a polynomial-time verifier. The first prover is trying to convince the verifier that the string is in the language, and the second prover is trying to convince the verifier that the string is not in the language. If the input x belongs to L, then the first prover can give an irrefutable proof y of this fact, i.e., the verifier will accept irrespective of the proof given by the second prover. Similarly, if the string does not belong to the language, then the second prover can furnish an irrefutable proof.

3 Key Lemma

In this section we show that if SAT does not have polynomial-size circuits, then for every k there exist polynomially many formulas such that every circuit of size n^k is wrong on at least one of these formulas.

Throughout this paper, we assume without loss of generality that if a circuit says that a formula is satisfiable, then it outputs a satisfying assignment. Thus the circuit can make errors on only one side. This implies that the language $\{\langle C, 1^n\rangle \mid C \text{ is wrong on a formula of size } n\}$ is in NP.

Lemma 3.1 *Fix $n > 0$. For every $k > 0$, if SAT does not have n^{k+2}-size circuits at length n, then there exists a set S of satisfiable formulas of length n, called* counter-examples*, such that every circuit of size n^k is wrong on at least one formula from S. The cardinality of S is polynomial in n.*

Proof. The proof uses ideas from Bshouty et al. [BC$^+$96]. We define a probabilistic process and show that if SAT does not have n^{k+2}-size circuits, then the probabilistic process outputs a set of counter-examples with nonzero probability. We build the set S of counter-examples in stages. At stage zero, S contains an arbitrary satisfiable formula. At each stage we add a formula to the set. Therefore, after $i-1$ stages, S has i counter-examples. We now describe stage i. Fix $m = 36n$.

Let T_i be the set of all n^k-size circuits that are correct on S. If T_i is empty, then we are done; so assume T_i is not empty. Uniformly and independently pick m circuits $c_1, c_2, \cdots, c_m$ from T_i. Let C be a circuit that takes majority vote of $c_1, \cdots, c_m$. Note that the size of C is at most n^{k+2}. Since SAT does not have n^{k+2}-size circuits, there exists a satisfiable formula ϕ on which C is not correct. Add ϕ to S. This completes stage i.

We claim that after polynomially many stages, T_i is empty. Thus S contains polynomially many formulas such that every circuit of size n^k is wrong on at least one formula in S.

Claim 3.2

$$Pr[\|T_{i+1}\| \leq 2/3\|T_i\|] > 0.$$

Proof. Denote the set of randomly chosen circuits by U. Given a formula ρ, let V_ρ be the set of all circuits in T_i that are correct on ρ. Call a formula ρ "bad" if $\|V_\rho\| > 2/3\|T_i\|$. In the following, we fix a bad ρ.

For $1 \leq i \leq m$, define random variables X_i as follows: $X_i = 1 \Leftrightarrow c_i \notin V_\rho$. Since c_i-s are picked independently and uniformly, $Pr[X_i = 1] = p \leq \frac{1}{3}$ for every i, $1 \leq i \leq m$. We note that since $p \leq \frac{1}{3}$,

$$Pr[\|U \cap V_\rho\| \leq \frac{1}{2}\|U\|] \leq Pr\left[\frac{\Sigma_{i=1}^m X_i}{m} - p > \frac{1}{6}\right].$$

Applying the Chernoff bound [Gol01, page 11] on the right hand side, we can show that

$$Pr[\|U \cap V_\rho\| \leq \frac{1}{2}\|U\|] \leq 2e^{-m*(1/18)} < 1/2^{2n}.$$

Since there can be at most 2^n bad formulas,

$$Pr[\exists \text{ bad } \rho \text{ such that } \|U \cap V_\rho\| \leq \frac{1}{2}\|U\|] < 1/2^n. \quad (1)$$

Consider the counter-example ϕ generated during stage i. Since ϕ is a counter-example to C, the majority circuit of $c_1, \cdots c_m$, more than $m/2$ circuits in U are wrong on ϕ. However, if this ϕ were a bad formula, then by Equation 1, with high probability, more than half the circuits from $U = \{c_1, \cdots, c_m\}$ would be correct on ϕ. It follows that the probability that ϕ is not bad is nonzero. Thus $\|V_\phi\| \leq 2/3\|T_i\|$ with high probability. Note that every circuit in T_{i+1} should be correct on ϕ. Thus it follows that $\|T_{i+1}\| \leq 2/3\|T_i\|$ with nonzero probability. This proves Claim 3.2. □

Therefore, after each stage, with nonzero probability, the number of circuits that are correct on S are reduced by a constant fraction. So after polynomially many stages all the n^k-size circuits would be wrong on S. Since we increase the size of S by one during each stage, the cardinality of S is bounded by a

polynomial. □

We also note that the above process can be implemented by a probabilistic polynomial-time-bounded machine that uses SAT as an oracle. At any stage we need the ability to pick circuits $c_1, c_2, \cdots, c_m$ uniformly at random from T_i, and generate a counter example ϕ to C where C is the circuit that takes majority vote of $c_1, \cdots, c_m$. The later task can be done by making queries to the following NP language.

$\{\langle C, x\rangle \mid \exists$ a satisfiable formula ϕ such that x is a prefix of ϕ and C is wrong on $\phi\}$.

Also note that once we obtain the counterexample ϕ, we can compute a satisfying assignment of ϕ using SAT as an oracle. So we can assume that S consists of satisfiable formulas along with the assignments. Now

$$T_i = \{C \mid C \text{ is a } n^k\text{-size circuit that is correct on } S\}.$$

Since S consists of satisfiable formulas along with the assignments, T_i is a set in P. Jerrum, Valiant, and Vazirani [JVV86] showed that picking elements, in an approximately uniform manner, from a set in P can be done in polynomial-time using SAT as an oracle. Using their procedure we can pick circuits from T_i in an approximately uniform manner.

4 Application to Two Queries

In this section we show an application of our lemma to the two queries problem.

Theorem 4.1 *If* $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, *then* $\mathrm{PH} = \mathrm{S}_2^p$.

To prove Theorem 4.1 we need the following theorem by Buhrman and Fortnow [BF99].

Theorem 4.2 (Buhrman-Fortnow) *If* $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, *then there exists a polynomial-time predicate* R *and a constant* $k > 0$ *such that for every* n, *one of the following holds.*

1. *Locally* $\mathrm{NP} = \text{co-NP}$*: For every unsatisfiable formula* ϕ *of length* n, *there is a short proof of unsatisfiability* w, *i.e.,* $\phi \notin \mathrm{SAT} \Leftrightarrow \exists w R(\phi, w)$, *where* $|w|$ *is bounded by a fixed polynomial in* n.

2. *There exists a circuit of size n^k that decides SAT at length n.*

We first show that if $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, then $\Sigma_2^p = \Pi_2^p$. We use Lemma 3.1 to decide whether locally $\mathrm{NP} = \mathrm{co\text{-}NP}$ or SAT has small circuits.

Lemma 4.3 *If $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, then $\Sigma_2^p = \Pi_2^p$.*

Proof. Let L be any language in Π_2^p. For any input x, the following holds:

$$x \in L \Leftrightarrow \forall y\ \phi_y \in \mathrm{SAT}.$$

Let $|\phi_y| = m$. By Theorem 4.2, if SAT does not have m^{k+2}-size circuits at length m, then every unsatisfiable formula of length m has a short proof of unsatisfiability.

We describe an NP machine with SAT as an oracle that accepts L. Recall that the set $\{\langle C, 1^n\rangle \mid C \text{ is wrong on a formula of length } n\}$ is in NP.

Consider the following machine M:

1. Guess 0 or 1

2. If the guessed bit is 0, guess a circuit C of size m^{k+2}, and ask the SAT oracle if C is a correct circuit for SAT at length m. If the answer is "no", then reject the input. If the answer is "yes", then C is a correct circuit for SAT at length m. This can be used to decide x, by asking the SAT oracle whether there is a y such that $C(\phi_y) = 0$. If the answer is "yes", then x does not belong to L; otherwise, x belongs to L.

3. If the guessed bit is 1, guess l satisfiable formulas $\phi_1, \cdots, \phi_l$ and ask the SAT oracle whether there is a circuit of size at most m^k that is correct on all the guessed formulas. (Note that l is the number of counter-examples obtained from Lemma 3.1.) If the answer is "yes", then reject the input. If the answer is "no", then there is no circuit (for SAT) of size m^k at length m. In this case, by Theorem 4.2, there is a polynomial-time predicate R such that for every unsatisfiable formula of length m, there is a short proof w. Ask the SAT oracle if x is in the following set:

$$\{x \mid \exists y \exists w R(\phi_y, w)\}.$$

   If x is in this set, then reject x, otherwise accept x.

We claim that the above algorithm is correct. Let $x \in L$. We consider the following two cases.

Case 1: SAT has m^{k+2}-size circuits at length m. In this case there exists a path of M that guesses the correct circuit and the machine accepts along this path.

Case 2: SAT does not have m^{k+2}-size circuits at length m. In this case, by Lemma 3.1, there exists a set of satisfiable formulas $\phi_1 \cdots \phi_l$ such that every circuit of size m^k is wrong on at least one of the formulas. Therefore, there is a path of M that correctly guesses these $\phi_1, \cdots, \phi_l$. Along this path M knows that $\mathrm{NP} = \text{co-NP}$ locally. So M accepts x along this path.

Next we show that if x does not belong to L, then every path of the machine rejects x. Again we treat two cases.

Case 1: SAT has m^{k+2}-size circuits at length m. Consider the paths that guessed 0 in the first step. The path that correctly guesses the circuit rejects. The paths that guess a wrong circuit also reject. Now, consider that paths the guessed 1. In this case, there may or may not exist a set of counter-examples against m^k-size circuits. If there are no counter-examples, then all paths reject. If there are counter-examples, then some paths will guess the correct counter-examples. However, the existence of counter-examples to m^k-size circuits implies that SAT does not have m^k-size circuits at length m. Thus by Theorem 4.2, locally $\mathrm{NP} = \text{co-NP}$. Thus all these paths correctly decide that $x \notin L$.

Case 2: SAT does not have m^{k+2}-size circuits at length m. In this case all the paths that guessed 0 in the first step reject. Consider the paths that guessed 1. By Lemma 3.1, there exists a set of counter-examples. The path that correctly guesses the counter-examples realizes that locally $\mathrm{NP} = \text{co-NP}$, and rejects x. The paths that guess wrong counter-examples also reject.

Therefore, M decides L. This shows that $\Sigma_2^p = \Pi_2^p$. □

Theorem 4.1 follows from Lemma 4.3 and the following lemma.

Lemma 4.4 *If* $\mathrm{P}^{\mathrm{NP}[1]} = \mathrm{P}^{\mathrm{NP}[2]}$, *then* $\Sigma_2^p \cap \Pi_2^p = \mathrm{S}_2^p$.

Proof. Let L be in $\Sigma_2^p \cap \Pi_2^p$. Thus

$$x \in L \Rightarrow \exists y\ \phi_y \notin \mathrm{SAT}\ \wedge\ \forall z\ \rho_z \in \mathrm{SAT}.$$

$$x \notin L \Rightarrow \exists z\ \rho_z \notin \mathrm{SAT}\ \wedge\ \forall y\ \phi_y \in \mathrm{SAT}.$$

Without loss of generality, assume that $|\phi_y| = |\rho_z| = m$. By Theorem 4.2, at length m either every unsatisfiable formula has a short proof of unsatisfiability, or there is a m^k-size circuit that decides SAT at length m.

In the former case, i.e., if every unsatisfiable formula has a short proof of satisfiability, the first prover's proof consists of y, ϕ_y, and a proof that ϕ_y is not satisfiable. And the second prover's proof consists of z, ρ_z, and a proof that ρ_z is not satisfiable.

In the later case, the first prover's proof consists of y, ϕ_y, and a circuit of size m^k. The second prover's proof consists of z, ρ_z, and a circuit of size m^k.

Upon receiving the proofs, the verifier executes the following algorithm. If either prover claims a short proof of unsatisfiability, then the verifier first checks whether the given short proof really proves that the formula in consideration (ϕ_y or ρ_z) to be unsatisfiable. The verifier accepts if the first prover's proof is correct and rejects if the second prover's proof is correct. Note that both of them cannot be correct.

Consider the case where both the provers give circuits. Here, the first prover is claiming that ϕ_y is unsatisfiable, and the second prover is claiming that ρ_z is unsatisfiable. Also, the first prover is implicitly claiming that for every z, ρ_z is satisfiable. Therefore, if the first prover is correct, then his circuit should be able to output a satisfying assignment of ρ_z given by the second prover. The verifier checks whether that is the case. The verifier accepts only if the first prover's circuit produces a satisfying assignment on ρ_z.

It is clear that the prover who gives a correct proof can convince the verifier. Therefore, L is in S_2^p. □

5 Further Work

It would be interesting to see whether more applications of Lemma 3.1 can be found. Can we improve the collapse in Theorem 4.1 to P^{NP}? What consequences can be obtained if one assumes $\mathrm{P}^{\mathrm{NP}[k]} = \mathrm{P}^{\mathrm{NP}[k+1]}$ for $k \geq 2$?

6 Acknowledgments

The authors thank Richard Chang for his comments on an earlier version of this paper. The third author thanks Alan Selman for his helpful insights and valuable

suggestions.

References

[BCO93] R. Beigel, R. Chang, and M. Ogihara. A relationship between difference hierarchies and relativized polynomial hierarchies. *Mathematical Systems Theory*, 26(3), pp. 291–310, 1993.

[BC+96] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3), pp. 421–433, 1996.

[BF99] H. Buhrman and L. Fortnow. Two queries. *Journal of Computer and System Sciences*, 59(2), pp. 182–194, 1999.

[Cai01] Jin-Yi Cai. $\mathrm{S}_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *Proceedings of the 42nd IEEE Conference on Foundations of Computer Science (FOCS)*, pp. 620–629, 2001.

[Can96] R. Canetti. More on BPP and the polynomial-time hierarchy. *Information Processing Letters*, 57(5), pp. 237–241, 1996.

[CK96] R. Chang and J. Kadin. The boolean hierarchy and the polynomial hierarchy; A closer connection. *SIAM Journal on Computing*, 25(2), pp. 340–354, 1996.

[Gol01] O. Goldreich. *Foundations of Cryptography – Volume 1*. Cambridge University Press, New York, 2001.

[JVV86] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(1986), pp. 169–188.

[Kad88] J. Kadin. The polynomial-time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 17(1988), pp. 1263–1282.

[Kre88] M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(1988), pp. 490–509.

[RS98] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Journal of Computational Complexity*, 7(2), pp. 152–162, 1998.

[Wag87] K. Wagner. Number-of-query hierarchies. Technical Report 158, Institut für Mathematik, Universitat Augsburg, October 1987.

[Wag89] K. Wagner. Number-of-query hierarchies. Technical Report 4, Institut für Informatik, Universitat Würzburg, February 1989.