Satisfiability with Exponential Families

Dominik Scheder and Philipp Zumstein

Institute of Theoretical Computer Science, ETH Zürich
8092 Zürich, Switzerland
dscheder@inf.ethz.ch, zuphilip@inf.ethz.ch

Abstract. Fix a set $S \subseteq \{0,1\}^*$ of exponential size, e.g. $|S \cap \{0,1\}^n| \in \Omega(\alpha^n), \alpha > 1$. The S-SAT problem asks whether a propositional formula F over variables $v_1, \ldots, v_n$ has a satisfying assignment $(v_1, \ldots, v_n) \in \{0,1\}^n \cap S$. Our interest is in determining the complexity of S-SAT. We prove that S-SAT is NP-complete for all context-free sets S. Furthermore, we show that if S-SAT is in P for some exponential S, then SAT and all problems in NP have polynomial circuits. This strongly indicates that satisfiability with exponential families is a hard problem. However, we also give an example of an exponential set S for which the S-SAT problem is not NP-hard, provided P $\neq$ NP.

Keywords: satisfiability, context-free grammars, VC-dimension, NP-hardness, polynomial circuits

1 Introduction

Given a set $S \subseteq \{0,1\}^*$ of all assignments, the S-SAT problem asks whether for a formula F over n variables there is an assignment $x \in S_n := S \cap \{0,1\}^n$ that satisfies F (F is then called S-satisfiable). The other assignments $\{0,1\}^n \setminus S$ can be seen as assignments which are a priori forbidden. If $|S_n|$ is polynomial in n and S_n can be enumerated in polynomial time then S-SAT is in P. To exclude this case we concentrate on *exponential families*, which are defined next.

Definition 1.1. *A monotonically increasing sequence $Q = (n_j)_{j \in \mathbb{N}} \subseteq \mathbb{N}$ has* polynomial gaps *if there is a polynomial $p(n)$ such that*

$$n_{j+1} \leq p(n_j)$$

for all $j \in \mathbb{N}$.

For example, define $n_j = 2^j$. Then $n_{j+1} = 2n_j$, so $p(n) := 2n$ shows that this sequence has polynomial gaps. This means, a sequence (n_j) can increase exponentially in j and still have polynomial gaps. Note that we can always assume w.l.o.g. that $p(n)$ is strictly increasing.

Definition 1.2. *The family $(S_n)_{n \geq 0}$ is called* exponential *if there exists $\alpha > 1$ and a sequence Q with polynomial gaps such that*

$$\forall n \in Q : |S_n| \geq \alpha^n \,.$$

We also say $S = \bigcup_{n \geq 1} S_n$ has exponential size.

For example families with $|S_n| \in \Omega(\alpha^n)$ are exponential (but we additionally allow to have some "gaps"). There are some subtleties involved with the definition of S-SAT. First, observe that we can interpret $x \in \{0,1\}^*$ as a truth assignment only if V, the set of variables, is *ordered*. Second, we require that V is given explicitly as a part of the input together with the formula F. To see why this is necessary, define $S_n = \{x \in \{0,1\}^n \mid x_n = 0\}$. Then the formula $v_1 \wedge v_2 \wedge v_3$ with $V = (v_1, v_2, v_3)$ is not S-satisfiable, but with $V = (v_1, v_2, v_3, v_4)$ it is. Note that we do not require every variable in V to occur in F (this does not affect our results but turns out to be a useful convention). For simplicity of notation, we agree that the variables of V are named $v_1, \ldots, v_n$, in this order. Finally we want to point out that S is some fixed language and therefore it is not part of the input.

The question whether S-SAT is NP-hard for all exponential S was first stated by Cooper [1] on his web page, though we are working with a more general notion of *exponential*. As far as we know, there have not been any further considerations about S-SAT neither by Cooper nor by anybody else.

Our Results

We prove that S-SAT is NP-complete for all exponential S that are context-free (Section 4). Further, we show that if S-SAT is in P for some exponential S, then SAT, and thus every problem in NP, has polynomial circuits (Section 5). This would imply that the polynomial hierarchy collapses to its second level [2]. Since this is widely believed to be false, it is a strong indication that S-SAT is a hard problem in general. However, we construct an exponential S such that S-SAT is not NP-hard, provided $\mathsf{P} \neq \mathsf{NP}$ (Section 6).

2 Some Observations

It is easy to show NP-hardness of S-SAT for $S_n = \{x \in \{0,1\}^n \mid x_1 = 0\}$ (and similar simple families): Let the formula F be an instance of SAT. We construct the formula F' which is identical to F but with every occurrence of x_1 replaced by $\bar{x}_1$ and vice versa. The formula F is satisfiable iff the formula $F \vee F'$ is S-satisfiable. This is a polynomial reduction from SAT to S-SAT.

If we view S itself as a language over the alphabet $\{0,1\}$, and therefore as a decision problem, we get the following connection:

Proposition 2.1. *S can be reduced to S-SAT in polynomial time.*

Proof. Given some $x = (x_1, \ldots, x_n) \in \{0,1\}^n$. Write $v^1 := v$ and $v^0 := \bar{v}$, respectively. Then x is the unique assignment in $\{0,1\}^n$ that satisfies the formula $F_x := v_1^{x_1} \wedge v_2^{x_2} \wedge \cdots \wedge v_n^{x_n}$ over V with $|V| = n$. Hence, F_x is S-satisfiable if and only if $x \in S_n$. Clearly, this is a polynomial reduction from S to S-SAT. □

Hence, S-SAT can have arbitrarily high complexity; it may even be undecidable. The next proposition demonstrates how we can employ a fast S-SAT algorithm, if existent, to solve SAT in significantly less than 2^n steps. We write $O^*(f(n))$ if we neglect polynomial factors.

Proposition 2.2. *Suppose there is some S with $|S_n| \geq \alpha^n$ for $\alpha > 1$ and all sufficiently large n. If S-SAT can be decided in time $O^*(\beta^n)$, then there is a randomized Monte Carlo algorithm for* SAT *with running time $O^*((2\beta/\alpha)^n)$.*

Proof. Let F be a satisfiable formula over a set V of variables, and let x be a satisfying assignment. For each variable $v \in V$, *switch* v with probability $1/2$, i.e. invert all its occurrences in F and its value according to the assignment x, resulting in a new formula F' and a new assignment x'. The assignment x' satisfies F' if and only if x satisfies the original formula F. Moreover, x' is uniformly distributed over $\{0,1\}^n$. Therefore, with probability $p := \Pr[x' \in S_n] \geq (\alpha/2)^n$ the formula F' is S-satisfiable. This can be tested in time $O^*(\beta^n)$. After repeating this process $(2/\alpha)^n$ times, the probability that at least one of the randomly generated formulas is S-satisfiable, is at least $1 - 1/e$, hence constant. On the other hand, if F is unsatisfiable, it will not become satisfiable by switching. We therefore have a Monte Carlo algorithm with running time $(2/\alpha)^n O^*(\beta^n)$. □

There are no known algorithms for SAT running in time $O^*(\gamma^n)$ for $\gamma < 2$, not even randomized ones. Proposition 2.2 with $\beta < \alpha$, therefore, is a first indication that S-SAT is a difficult problem.

In fact, the currently best known deterministic algorithm for 3-SAT (satisfiability of formulas in *conjunctive normal form* where every disjunction consists of at most 3 literals) can be viewed as a derandomized version of the randomized algorithm in the proof of Proposition 2.2: Let the *Hamming distance* $d(x, y)$ of two vectors $x, y \in \{0,1\}^n$ be the number of bits in which they differ. The *Hamming Ball* of radius r around x is, in analogy to the usual definition of a ball, the set $B_r(x) := \{y \in \{0,1\}^n \mid d(x,y) \leq r\}$. We look at the family $S_n = B_{\rho n}(\mathbf{0})$ where ρ is some constant. Then

$$|S_n| = \sum_{i=0}^{\rho n} \binom{n}{i} \approx 2^{H(\rho)n}\,, \quad H(t) = -t \log t - (1-t)\log(1-t)\,.$$

Therefore $S = (S_n)_{n \geq 0}$ is an exponential family. For 3-CNFs, S-SAT can be decided in $3^{\rho n}$ steps (by splitting on 3-clauses), which for appropriately chosen ρ is much smaller than $2^{H(\rho)n}$. By choosing many Hamming balls centered at different points (randomly) and by choosing the optimal value of ρ this yields an algorithm deciding 3-SAT in $O^*(1.5^n)$ steps. Note that choosing a random point as center of the Hamming ball is equivalent to switching the formula randomly and keeping the Hamming ball centered at $(0, \ldots, 0)$ all time. It takes some additional effort to derandomize the algorithm. For details, see Dantsin et al. [3].

3 *S*-SAT and the VC-dimension

To obtain a systematical way of proving NP-hardness of S-SAT (if possible), we exploit the notion of *shattering* and the *Vapnik-Chervonenkis-dimension* $d_{\mathrm{VC}}(S_n)$, short VC-dimension, of a set $S_n \subseteq \{0,1\}^n$. These concepts were first introduced by Vapnik and Chervonenkis [4]. Let V with $|V| = n$ be an ordered set of variables. We say $I \subseteq [n]$ is *shattered* by S_n if any assignment to $V_I := \{v_i \mid i \in I\}$ can be realized by S_n. Formally, for every $x \in \{0,1\}^{|I|}$ there is a $y \in S_n$ with $y|_I = x$, where $y|_I$ denotes the $|I|$-bit vector $(y_i)_{i \in I}$. The VC-dimension d_{VC} is the size of a largest shattered set. Obviously, $0 \leq d_{\mathrm{VC}}(S_n) \leq n$. The intuition is that large sets have large VC-dimensions. This is quantified by the following lemma, which was proven several times independently, e. g. by Sauer [5].

Lemma 3.1. *Suppose $d_{\mathrm{VC}}(S_n) \leq d \leq n/2$. Then*

$$|S_n| \leq \sum_{i=0}^{d} \binom{n}{i} \leq 2^{H(\frac{d}{n})n}$$

where $H(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function.

Corollary 3.2. *Suppose $S \subseteq \{0,1\}^*$ has exponential size. Then there is a polynomial $q(n)$ such that for each $n \in \mathbb{N}$ there exists $N \leq q(n)$ and an index set $I \subseteq [N]$ with $|I| \geq n$ such that I is shattered by S_N.*

Proof. Let $(n_j)_{j \in \mathbb{N}}$ be the sequence with polynomial gaps corresponding to the exponential family S, i.e. there is an $\alpha > 1$ and a polynomial $p(n)$ such that $n_{j+1} \leq p(n_j)$ and $|S_{n_j}| \geq \alpha^{n_j}$ for all j. Choose $\delta \in (0, 1/2]$ such that $H(\delta) = \log \alpha$ and k such that $n_k \leq \frac{n}{\delta} \leq n_{k+1} =: N$. By Lemma 3.1, $d_{\mathrm{VC}}(S_N) \geq \delta N \geq n$, so there exists a shattered set $I \subseteq [N]$ with $|I| \geq n$. Note that $N = n_{k+1} \leq p(n_k) \leq p(n/\delta) =: q(n)$, as required. □

Although we know that a large shattered set exists, it is not clear how we can compute it efficiently. Let us for the moment assume that we can. Then there is a polynomial reduction from SAT to S-SAT:

Theorem 3.3. *Let $S \subseteq \{0,1\}^*$ be of exponential size and let $p(n)$ be a polynomial. Suppose that for all n, we can compute, in time polynomial in n, some number $N \leq p(n)$ and some index set $I \subseteq [N]$ with $|I| \geq n$ that is shattered by S_N. Then S-SAT is NP-hard.*

Proof. The existence of such a I is guaranteed by Corollary 3.2. Suppose it can be computed efficiently. Let F be a formula over the variables $V_n = \{v_1, \ldots, v_n\}$. We construct a new formula F' over V_N by renaming each v_j occurring in F into v_{i_j} where $I \supseteq \{i_1, \ldots, i_n\}$. We claim that F is satisfiable iff F' is S-satisfiable. Suppose $x \in \{0,1\}^n$ satisfies F. Clearly, there is some $x' \in \{0,1\}^N$ satisfying F', since F and F' differ only in the names of their variables. Every assignment

$y \in \{0,1\}^N$ that agrees with x' in the variables $(v_{i_1}, \ldots, v_{i_n})$ also satisfies F'. It follows from the definition of shattering that S_N contains such a y. Hence, F' is S-satisfiable. The reverse direction is clear. This polynomial reduction shows that S-SAT is NP-hard, under these conditions. □

Why does this method not work general? The difficulty is that we do not know which subset of variables is shattered, we only know that there is one. It is also futile to try to compute a large shattered set directly from S_n, since a polynomial reduction cannot deal with S_n explicitly, as $|S_n|$ is exponential in n (at least the S_n we are interested in is). Note that the brute force approach to computing the VC-dimension of a set will take time *polynomial* in $|S_n|$, if $|S_n|$ is exponentially in n. This is in contrast to the result of Papadimitriou and Yannakakis [6] that computing the VC-dimension of an explicitly given S_n (of size not necessarily exponential in n) is LOGNP-complete, hence unlikely to be in P. But this is no help to us: though computing the VC dimension takes time polynomial in $|S_n|$, observe that $|S_n|$ is itself exponential in n.

We see that a polynomial reduction from SAT to S-SAT must somehow have certain implicit knowledge of $S = \cup S_n$. One way would be a (regular, context-free, ...) grammar of S (if there is one).

Theorem 3.4. *If $S \subseteq \{0,1\}^*$ is a regular language and $S_n := S \cap \{0,1\}^n$, then $d_{\mathrm{VC}}(S_n)$ and a shattered set $I \subseteq [n]$ of this size can be computed in $O(n^2)$ (where the hidden constant factor is doubly exponential in the size of the regular grammar).*

The proof of this theorem is quite technical and is therefore omitted here. Instead, we will prove a similar theorem for context-free languages where we do not insist on computing a *largest* shattered index set, but only a *sufficiently large* one.

4 NP-Completeness of Context-Free S-SAT

In this section, we prove that S-SAT is NP-complete if S is a context-free language and has exponential size. It suffices to show how to find a large shattered index set. To be more precise, for any given n, we will find some $N \in O(n)$ and $I \subseteq [N]$ with $|I| \geq n$ such that I is shattered by S_N. In combination with the results from Section 3, this proves NP-hardness. It is clear that S-SAT is in NP if S is context-free, since deciding whether $x \in S$ and verifying that x is satisfying can be done in polynomial time.

In the following, we denote the nonterminal symbols appearing in the context-free grammar for S by upper case letters $S_0, A, B, C, \ldots$, where S_0 is the starting symbol. The only terminal symbols are $0, 1$. All rules in a context-free grammar are of the form $A \vdash w$ for a word w possibly containing nonterminals. $A \vdash^* w$ means that w can be derived from A in finitely many steps. Finally, the length

of a word x is denoted by $|x|$.

Let S be a context-free, exponential language which is generated by the grammar G. All calculations on the grammar can be done in advance and therefore do not contribute to the running time. In particular, we may assume that G does not contain *useless* nor *unreachable* nonterminal symbols, i.e. for every nonterminal A, we have $A \vdash^* x$ for some $x \in \{0,1\}^*$, and $S \vdash^* w$ for some w with $A \in w$. We call such a grammar *reduced*. For a nonterminal A, define

$$\ell(A) := \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^* : A \vdash^* xAy\} ,$$
$$r(A) := \{y \in \{0,1\}^* \mid \exists x \in \{0,1\}^* : A \vdash^* xAy\} .$$

Call some $X \subseteq \{0,1\}^*$ *commutative* if $xy = yx$ for all $x, y \in X$.

Lemma 4.1 (Ginsburg [7], Theorem 5.5.1). *Let G be a reduced context-free grammar and let $L(G)$ be the language generated by G. Then $|L(G) \cap \{0,1\}^n|$ is polynomial in n if and only if for every nonterminal A, $\ell(A)$ and $r(A)$ are commutative.*

Theorem 4.2. *Suppose $S \subseteq \{0,1\}^*$ has exponential size and is a context-free language. Then S-SAT is* NP*-complete.*

Proof. We will show how to compute large shattered sets, for every n. Let G be a reduced context-free grammar for S. Since S has exponential size, $|S_n|$ is surely not polynomial in n. Therefore, Lemma 4.1 implies that there is a nonterminal A such that $\ell(A)$ or $r(A)$ is not commutative. Suppose w.l.o.g. that $\ell(A)$ is not commutative, and let $x_1, x_2 \in \ell(A)$ such that $x_1x_2 \neq x_2x_1$. Hence, there is a position i such that w.l.o.g. $(x_1x_2)_i = 0$ and $(x_2x_1)_i = 1$. By definition, there are $y_1, y_2 \in \{0,1\}^*$ such that $A \vdash^* x_1Ay_1$ and $A \vdash^* x_2Ay_2$. By applying k times either $A \vdash^* x_1x_2Ay_2y_1$ or $A \vdash^* x_2x_1Ay_1y_2$, we can create arbitrary 0s and 1s at the positions $i + k \cdot |x_1x_2|$ for any k. In order to reach A from S_0, we use $S_0 \vdash^* aAb$, and in the end we use $A \vdash^* w$ to obtain a word in $\{0,1\}^*$ for some $a, b, w \subseteq \{0,1\}^*$. Hence if we set $N := |a| + |b| + |w| + n(|x_1x_2| + |y_1y_2|)$, then $I := \{|a| + k|x_1x_2| + i \,:\, 0 \leq k \leq n-1\}$ is of size n, and it is shattered by S_N. All these calculations can be done in time $O(n)$ and N is linear in n. Thus, by Theorem 3.3, S-SAT is NP-hard. Since S-SAT $\in$ NP, it is NP-complete. □

5 *S*-SAT and Polynomial Circuits

In the previous section, we have seen that if we can efficiently compute large shattered sets, then S-SAT is NP-hard. If we cannot compute those sets, then we do not have a systematic way of proving NP-hardness (although there are simple examples where large shattered sets of S_n cannot be computed at all, and still S-SAT is NP-hard). However, we will prove a result that is "almost as good" as proving NP-completeness: if S-SAT is in P for some exponential S, then SAT has polynomial circuits.

Since boolean circuits are standard terminology in complexity theory, we do not give a formal definition. Furthermore, because we are interested in the *size* of a circuit, i.e. the number of its gates, and not in the *depth*, it does make a difference whether we allow bounded fan-in or not. For an overview of boolean circuits in complexity theory, see [8].

Definition 5.1. *A* circuit family *is a sequence $\mathcal{C} = (C_1, C_2, \dots)$ of boolean circuits, where each C_n has n input gates. If each C_n has exactly one output gate, then $\mathcal{C}$ computes a function $f : \{0,1\}^* \to \{0,1\}$, or equivalently, decides a language $L \subseteq \{0,1\}^*$.*

If the size of C_n grows polynomially in n, then $\mathcal{C}$ is a polynomial circuit family*. If there exists an algorithm that computes and outputs C_n in time polynomial in n, we call $\mathcal{C}$ a* uniform polynomial circuit family*.*

It is not hard to show that a language $L \in \{0,1\}^*$ can be decided by uniform polynomial circuits if and only if it is in P. There are even undecidable languages with (nonuniform, of course) polynomial circuits. However, there are good reasons to believe that NP-complete problems do not have polynomial circuits, whether uniform or not: Karp and Lipton [2] showed that if NP-complete problems have polynomial circuits, then the polynomial hierarchy collapses to its second level. The connection to S-SAT is immediate:

Theorem 5.2. *If S-SAT is in P for some exponential S, then* SAT *has (possibly nonuniform) polynomial circuits.*

Proof. From Corollary 3.2, we know that for each n there exists an $N \leq q(n)$ and an index set $I \subseteq [N]$ with $|I| \geq n$ such that I is shattered by S_N. For each n, there is a boolean circuit of polynomial size that takes a formula F over n variables as input and outputs a formula F' over N variables, where F' is identical to F, but with the all variables from F replaced by variables in I. Note that the circuit *exists*, though it might not be constructible in polynomial time. By assumption, there is a second circuit of polynomial size deciding S-SAT for formulas with N variables. This circuit can be constructed in polynomial time. Combining these two circuits yields a polynomial circuit deciding SAT. □

It might be possible that NP has polynomial circuits and still $\mathsf{P} \neq \mathsf{NP}$. Hence, this result is weaker than proving NP-hardness for S-SAT in general.

6 Some *S*-SAT Which Is Not NP-hard

In this section we will prove—under reasonable assumptions—that there is an exponential S such that S-SAT is not NP-hard. We will use a classical tool of complexity theory: diagonalization. Let us first introduce some notation. As we stated in Section 1, we assume that an instance of S-SAT always comes with an explicitly given set of variables $V = \{v_1, \dots, v_n\}$. For a formula F, let $n(F)$ denote the size of this variable set, *not* the number of variables actually present in F. These sets can differ, as we have seen.

Definition 6.1. *A function φ mapping formulas to formulas is called a* SAT-reduction *if, for all satisfiable formulas F and unsatisfiable formulas F', we have $\varphi(F) \neq \varphi(F')$. If there exists an algorithm which computes φ in polynomial time, then we say that it is a* polynomial SAT-reduction.

Consider for example the mapping φ which maps every satisfiable formula to 1 and every unsatisfiable formula to 0. This φ is a SAT-reduction but it is not polynomial (provided $\mathsf{NP} \neq \mathsf{P}$). It should be clear that any function φ, that does not fulfill the condition of being a SAT-reduction, is disqualified from being a reduction from SAT to any S-SAT in the first place.

Theorem 6.2. *Provided that $\mathsf{P} \neq \mathsf{NP}$, there is an S with $|S_n| = 2^n$ for at least every second n, and* SAT $\not\leq_p$ S-SAT. *Thus, S has exponential size and S-*SAT *is not* NP*-hard.*

Provided that $\mathsf{P} \neq \mathsf{NP}$, we will show that there are arbitrarily large formulas having preimages that are satisfiable for every polynomial SAT-reduction φ. Nnote that F might have several preimages, but according to the definition of a SAT-reduction, they are either all satisfiable or all unsatisfiable. If such a formula F has n variables, and G is one of its satisfiable preimages, then setting $S_n = \emptyset$ prevents φ from being a polynomial reduction from SAT to S-SAT, since G is satisfiable but $F = \varphi(G)$ is not S-satisfiable. We then choose such n_i for each polynomial SAT-reduction φ_i and set $S_n = \{0,1\}^n$ for all other remaining values of n. By leaving gaps between the n_i, we guarantee that S has exponential size.

Lemma 6.3. *Provided that $\mathsf{P} \neq \mathsf{NP}$, then for every polynomial* SAT*-reduction φ, there are arbitrarily large formulas (in terms of $n(F)$) with satisfiable preimages.*

Proof. For the sake of contradiction, suppose that there is some SAT-reduction φ and some n_0 such that $n(\varphi(F)) \leq n_0$ for all satisfiable F. Consider

$$\mathcal{F}_0 := \{\varphi(F) \mid F \text{ is a satisfiable formula}\}$$

the image of all satisfiable formulas. By assumption, all formulas in $\mathcal{F}_0$ have no more than n_0 variables, implying that $\mathcal{F}_0$ is finite. Clearly, F is satisfiable iff $\varphi(F) \in \mathcal{F}_0$. Thus, φ reduces SAT to the finite language $\mathcal{F}_0$. Since every finite language is in P, SAT is in P, too. This contradicts our assumption. □

Proof (of Theorem 6.2). The Lemma gives us functions $n(\varphi, n_0)$, $F(\varphi, n_0)$ such that $n(\varphi, n_0) \geq n_0$, and $F(\varphi, n_0)$ has exactly $n(\varphi, n_0)$ variables and has satisfiable preimages.

Let $\varphi_1, \varphi_2, \ldots$ be an enumeration of all polynomial SAT-reductions (there are countably many) and define

$$\begin{aligned} n_1 &:= n(\varphi_1, 0)\,, \\ n_{i+1} &:= n(\varphi_{i+1}, n_i + 2)\,. \end{aligned}$$

$$S_n := \begin{cases} \emptyset & \text{if } n = n_i \text{ for some } i; \\ \{0,1\}^n & \text{otherwise.} \end{cases}$$

First, note that $n_{i+1} - n_i \geq 2$. Therefore, if $S_n = \emptyset$, then $|S_{n-1}| = 2^{n-1}$. Hence at least half of the levels are "full". Second, suppose some φ_i reduces SAT to S-SAT. By construction, there is a satisfiable formula F such that $\varphi_i(F)$ has exactly n_i variables. Unfortunately, S_{n_i} is empty, so $\varphi_i(F)$ is not S-satisfiable, hence φ_i is not a reduction, which is a contradiction. Since every SAT-reduction appears as some φ_i in our sequence, the proof is complete. □

This is nice, but has the drawback that S might have gaps, i.e. not every level has exponential size. The next construction gives us an S that overcomes this deficiency.

Theorem 6.4. *Provided that* $\mathsf{RP} \neq \mathsf{NP}$*, there is an* S *with* $|S_n| \geq 2^{n-1}$ *for all* n *such that* S-SAT *is not* NP*-hard.*

The problem above was that, in order to ensure that for the satisfiable formula $F = F(\varphi, n_0)$, $\varphi(F)$ is not S-satisfiable, we had to set $S_n = \emptyset$ for $n = n(\varphi, n_0)$, creating a "gap" in S. Alternatively, we could set $S_n := \{0,1\}^n \backslash \mathrm{sat}(\varphi(F))$, where $\mathrm{sat}(\varphi(F))$ is the set of all assignment which satisfy $\varphi(F)$. Clearly this suffices to ensure that $\varphi(F)$ is not S-satisfiable, preventing φ from being a reduction from SAT to S-SAT. If, in addition, $\mathrm{sat}(\varphi(F))$ is small, $|S_n|$ will be exponential in n. Let us now first focus on what happens when it is never small.

Definition 6.5. *A* SAT*-reduction* φ *is referred to* sharp*, if there is some* n_0 *such that for all* F *with* $n := n(\varphi(F)) \geq n_0$*, the following two statements hold:*

(i) F *and* $\varphi(F)$ *are* SAT*-equivalent, that is, either both are satisfiable, or both are not*
(ii) if $\varphi(F)$ *is satisfiable, then* $|\mathrm{sat}(\varphi(F))| > 2^{n-1}$

The choice of 2^{n-1} is arbitrary. Any number x with $x/2^n > \epsilon > 0$ and $2^n - x$ being exponential would be good as well. The image of a sharp reduction consists of formulas with at most n_0 variables, unsatisfiable formulas, and formulas with a huge number of satisfying assignments.

Lemma 6.6. *If there is a polynomial sharp* SAT*-reduction* φ*, then* $\mathsf{RP} = \mathsf{NP}$*.*

Proof. We give a randomized algorithm for SAT with a bounded error probability. Similar to the proof of Lemma 6.3, define

$$\mathcal{F}_0 := \{\varphi(F) \mid F \text{ is satisfiable and } n(\varphi(F)) \leq n_0\}$$

Again, this set is finite. We compute satisfiability of some input formula F with $n(F) = n$ as follows: if $n(\varphi(F)) \leq n_0$, we simply check whether $\varphi(F) \in \mathcal{F}_0$, which can be done in constant time. Otherwise, either both F and $\varphi(F)$ are unsatisfiable, or both are satisfiable, but then $\mathrm{sat}(\varphi(F))$ is huge. Let x be a uniformly at random chosen assignment out of $\{0,1\}^n$ for $n = n(\varphi(F))$ and return *satisfiable* if x satisfies $\varphi(F)$ and *unsatisfiable* otherwise. If F is unsatisfiable, the algorithm always answers correctly, otherwise the answer is wrong with a probability $p \leq 1/2$. Thus SAT is in RP, and hence $\mathsf{RP} = \mathsf{NP}$. □

The contrapositive of Lemma 6.6 reads as follows: Provided that $\mathsf{RP} \neq \mathsf{NP}$, no polynomial SAT-reduction φ is sharp, which means that for all φ, n_0, there exist $n = n(\varphi, n_0) \geq n_0$, $F = F(\varphi, n_0)$, such that $\varphi(F)$ has n variables and one of the following holds:

(i) F and $\varphi(F)$ are not SAT-equivalent
(ii) they are SAT-equivalent, $\varphi(F)$ is satisfiable, and $|\mathrm{sat}(\varphi(F))| \leq 2^{n-1}$

Proof (of Theorem 6.4). Using the function $n(\varphi, n_0)$ and our sequence $\varphi_1, \varphi_2, \ldots$ of polynomial SAT-reductions, we define

$$\begin{aligned} n_1 &:= n(\varphi_1, 0)\,, & F_1 &:= F(\varphi_1, 0)\,, \\ n_{i+1} &:= n(\varphi_{i+1}, n_i + 1)\,, & F_{i+1} &:= F(\varphi_{i+1}, n_i + 1)\,. \end{aligned}$$

So the F_i are the formulas with n_i variables provided by the contrapositive of Lemma 6.6, and the n_i are all distinct. If case (i) above applies to F_i, we say n_i is of type (i), if case (ii) applies, n_i is of type (ii). We define S by

$$S_n := \begin{cases} \{0,1\}^n \setminus \mathrm{sat}(\varphi_i(F_i)) & \text{if } \mathrm{n} = \mathrm{n_i} \text{ is of type (ii);} \\ \{0,1\}^n & \text{otherwise.} \end{cases}$$

We claim that every φ fails to be a reduction from SAT to S-SAT. Take any φ_i. If n_i is of type (i), then F_i and $\varphi_i(F_i)$ are not SAT-equivalent, and since $S_{n_i} = \{0,1\}^{n_i}$, $\varphi_i(F_i)$ is S-satisfiable iff F_i is not satisfiable. Thus, φ is not a reduction from SAT to S-SAT. If n_i is of type (ii), then F_i and $\varphi_i(F_i)$ are both satisfiable, but $\varphi_i(F_i)$ is not S-satisfiable, since $S_{n_i} = \{0,1\}^{n_i} \setminus \mathrm{sat}(\varphi_i(F_i))$. Hence φ_i fails also in this case. Finally, note that $|S_n| \geq 2^{n-1}$ for all n. □

As one referee pointed out, Theorem 6.2 looks like a weaker version of Ladner's theorem [9], which states that there are *intermediate* languages $L \in \mathsf{NP} \setminus \mathsf{P}$ which are not NP-complete, provided that $\mathsf{P} \neq \mathsf{NP}$. In fact, we could use Ladner's theorem to define a set $S \subseteq \{0,1\}^*$ such that S-SAT is an intermediate language. Unfortunately, it is not clear whether such S is exponential according to Definition 1.2. Certainly, it is much less "dense" than the languages S defined in the proofs of Theorem 6.2 and Theorem 6.4, for which it holds that for all n $|\bigcup_{i \leq n} S_i| \in \Omega(2^n)$ and $|S_n| \geq 2^{n-1}$, respectively.

7 Conclusion

Let us go back to where we started. We were interested in the complexity of S-SAT, for some given $S \subseteq \{0,1\}^*$. We can restate the question:

> **Problem:** Find a large natural class $\mathcal{S} \subseteq 2^{\{0,1\}^*}$ of sets of assignments, such that S-SAT is NP-hard (or even NP-complete) for all $S \in \mathcal{S}$.

As we have seen, the set of all exponential context-free languages is such a class, while the class of all exponential languages is not such a class. Might it be that S-SAT is NP-complete for all exponential S in P? In the light of Ladner's theorem [9], this seems unlikely.

References

1. Cooper, J.: Josh Cooper's Math Pages: Combinatorial problems I like `http://www.math.sc.edu/~cooper/combprob.html`.
2. Karp, R., Lipton, R.J.: Some connections between nonuniform and uniform complexity classes. In: Enseign. Math. 28. (1982) 191–201
3. Dantsin, E., Goerdt, A., E. A.H., Kannan, R., Kleinberg, J., Papadimitriou, C., Raghavan, O., Schöning, U.: A deterministic $(2 - 2/(k + 1))^n$ algorithm for k-SAT based on local search. In: Theoretical Computer Science 289. (2002) 69–83
4. Vapnik, V., Chervonenkis, A.: On the uniform convergence of relative frequencies of events to their probabilities. Theory Prob. Appl. **16** (1971) 264–280
5. Sauer, N.: On the density of families of sets. In: J. Comb. Theory, Ser. (A). Volume 13. (1973) 145–147
6. Papadimitriou, C.H., Yannakakis, M.: On limited nondeterminism and the complexity of the V-C dimension. J. Comput. Syst. Sci. **53**(2) (1996) 161–170
7. Ginsburg, S.: The Mathematical Theory of Context-Free Languages. McGraw-Hill, Inc., New York, NY, USA (1966)
8. Papadimitriou, C.: Computational Complexity. Addison Wesley (1994)
9. Ladner, R.E.: On the structure of polynomial time reducibility. J. ACM **22**(1) (1975) 155–171