High Sets for NP

Johannes Köbler* Uwe Schöning*

1 Introduction

We consider sets that are high for the complexity class $\mathcal{NP}$ with respect to several operators on complexity classes. The notion of lowness and highness in the context of complexity theory was introduced by Schöning [34] and further investigated and generalized under several aspects (see [24, 7, 9, 37, 22, 20, 27, 4, 31, 25, 26]).

Whereas lowness generalizes the class $\mathcal{P}$ and is incompatible with $\mathcal{NP}$-completeness (unless something unlikely like a collapse of the polynomial-time hierarchy follows), the concept of highness generalizes $\mathcal{NP}$-completeness (and $\mathcal{NP}$-hardness). We review several other generalized $\mathcal{NP}$-completeness and $\mathcal{NP}$-hardness notions with respect to weak reducibilities and compare it to respective highness classes. The main contribution of this paper is a proof that $\mathcal{NP}$-hard sets with respect to polynomial-size circuit reducibility are high for $\mathcal{NP}$ with respect to the operator $\mathcal{ZPP}(\mathcal{NP}(\ ))$.

Consider some operator $\mathcal{O}$ that maps languages to classes of languages. Let

$$\mathcal{O}(\mathcal{P}) = \bigcup_{A\in\mathcal{P}} \mathcal{O}(A) \quad \text{and} \quad \mathcal{O}(\mathcal{NP}) = \bigcup_{A\in\mathcal{NP}} \mathcal{O}(A).$$

If some language L has the property that $\mathcal{O}(L) \subseteq \mathcal{O}(\mathcal{P})$, then we call L a *low set* with respect to operator $\mathcal{O}$. If, on the other hand, $\mathcal{O}(\mathcal{NP}) \subseteq \mathcal{O}(L)$, then L is called a *high set* with respect to the operator $\mathcal{O}$, or $\mathcal{O}$-high, for short. As we will see, highness generalizes the concept of $\mathcal{NP}$-hardness; and if additionally $L \in \mathcal{NP}$ holds, highness generalizes the concept of $\mathcal{NP}$-completeness (of course, this concept of lowness and highness can be generalized to pairs of classes, other than $\mathcal{P}$ and $\mathcal{NP}$; but we will not pursue this here).

We use such operators $\mathcal{O}$ where we believe that $\mathcal{O}(\mathcal{P}) \neq \mathcal{O}(\mathcal{NP})$. Therefore, if some set A is both $\mathcal{O}$-low and $\mathcal{O}$-high then it follows $\mathcal{O}(\mathcal{NP}) \subseteq \mathcal{O}(A) \subseteq \mathcal{O}(\mathcal{P})$ which is considered to be unlikely. For example, if $\mathcal{O} = \Sigma_k^p$, a class of the

*Universität Ulm, Abt. Theoretische Informatik, D-89069 Ulm, Germany.
{koebler,schoenin}@informatik.uni-ulm.de

(relativized) polynomial-time hierarchy, then the existence of a Σ_k^p-low and Σ_k^p-high set implies a collapse of the polynomial-time hierarchy.

The following figure illustrates the relative position of low and high sets with respect to two different operators $\mathcal{O}_1$ and $\mathcal{O}_2$ of different strength, say $\mathcal{O}_1(L) \subseteq \mathcal{O}_2(L)$.

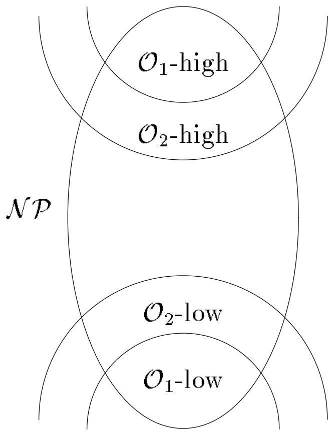

Figure 1: The location of low and high sets.

2 Reducibilities and Operators

We will define a variety of polynomial-time reducibilities that can be used to define different versions of $\mathcal{NP}$-completeness or $\mathcal{NP}$-hardness. We assume familiarity with the classes $\mathcal{P}$, $\mathcal{NP}$, $co\mathcal{NP}$, and the polynomial-time hierarchy (see [17]). The standard reducibility used to define $\mathcal{NP}$-completeness is the following.

Definition 2.1 *A set A is polynomial-time* many-one reducible *to a set B, denoted by $A \leq_m^{\mathcal{P}} B$, if there is a polynomial-time computable function f such that $A = f^{-1}(B)$.*

Already in Cook's seminal paper [15] a more general reducibility is used.

Definition 2.2 *A set A is polynomial-time* Turing (or Cook) reducible *to a set B, denoted by $A \leq_T^{\mathcal{P}} B$, if there is a polynomial-time deterministic oracle Turing machine M such that $A = L(M, B)$.*

Thus, a many-one reduction can be regarded as the special case of a Turing reduction where the oracle Turing machine M asks exactly one *positive query*, i.e., M accepts if and only if the answer to that query is positive. The (standard) notion of $\mathcal{NP}$-completeness (which is defined in terms of polynomial-time many-one reducibility) can be extended by using more general polynomial-time reducibilities, allowing a certain amount of nondeterminism, or randomness, or non-uniformity. We review those definitions which are relevant for this paper.

Definition 2.3 *A set A is* strongly nondeterministically Turing-reducible *to B in polynomial time, denoted by $A \leq_T^{\mathcal{SNP}} B$, if $A \in \mathcal{NP}(B) \cap co\mathcal{NP}(B)$ holds.*

This is a generalization of so-called γ-reducibility, which can be understood as a strong nondeterministic *many-one* reducibility. Several (number theoretic) problems have been shown to be γ-complete (hence $\leq_T^{\mathcal{SNP}}$-complete) and are not known to be $\mathcal{NP}$-complete in the usual sense [2, 3, 32, 33] (see also [14]). Long [30] has systematically studied these strong nondeterministic reducibilities.

It is easily seen that $A \leq_T^{\mathcal{SNP}} B$ if and only if $\mathcal{NP}(A) \subseteq \mathcal{NP}(B)$ (see [39, 30]). Therefore, our first comparison with highness immediately follows: $\leq_T^{\mathcal{SNP}}$-hardness is exactly the same as $\mathcal{NP}$-highness (see Figure 3).

A further idea is to introduce randomness in the reduction. Typical random algorithms introduce just one-sided errors. We take this into account in the following definitions which are straightforward translations from the respective complexity class definitions. (The original reducibilities introduced by Adleman and Manders [3] have been slightly modified to make them more robust, i.e. closed under composition; see [42, 13, 38]).

Definition 2.4

a) A set A is polynomial-time randomly Turing reducible *to a set B, denoted by $A \leq_T^{\mathcal{RP}} B$, if there is a polynomial-time deterministic oracle Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,*

$$\begin{aligned} x \in A \quad &\Rightarrow \quad M^B(x,r) = 1 \textit{ with probability } \geq 1/q(n), \\ x \notin A \quad &\Rightarrow \quad M^B(x,r) = 0 \textit{ with probability } = 1. \end{aligned}$$

Here and below, the string r is drawn uniformly at random from $\{0,1\}^{p(n)}$, for some polynomial p.

b) A set A is polynomial-time co-randomly Turing reducible *to a set B, denoted by $A \leq_T^{co\mathcal{RP}} B$, if $\overline{A} \leq_T^{\mathcal{RP}} B$, i.e., there is a polynomial-time deterministic oracle Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,*

$$\begin{aligned} x \in A \quad &\Rightarrow \quad M^B(x,r) = 1 \textit{ with probability } = 1, \\ x \notin A \quad &\Rightarrow \quad M^B(x,r) = 0 \textit{ with probability } \geq 1/q(n). \end{aligned}$$

By restricting the oracle Turing machine M to ask exactly one positive query we get definitions for the many-one versions of these reducibilities. Thus, a set A is polynomial-time *randomly many-one reducible* to a set B, denoted by $A \leq_m^{\mathcal{RP}} B$, if there is a polynomial-time deterministic Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,

$$\begin{array}{lll} x \in A & \Rightarrow & M(x,r) \in B \text{ with probability } \geq 1/q(n), \\ x \notin A & \Rightarrow & M(x,r) \notin B \text{ with probability } = 1. \end{array}$$

Similarly, A is polynomial-time *co-randomly many-one reducible* to a set B, denoted by $A \leq_m^{co\mathcal{RP}} B$, if $\overline{A} \leq_m^{\mathcal{RP}} B$, i.e., there is a polynomial-time deterministic Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,

$$\begin{array}{lll} x \in A & \Rightarrow & M(x,r) \in B \text{ with probability } = 1, \\ x \notin A & \Rightarrow & M(x,r) \notin B \text{ with probability } \geq 1/q(n). \end{array}$$

In [41] it is shown that $\mathsf{SAT} \leq_m^{\mathcal{RP}}$ $\mathsf{UniqueSAT}$ ($\mathsf{UniqueSAT}$ contains all Boolean formulas that have a unique satisfying assignment; in fact, SAT reduces to every solution A of the "promise problem" $(1\mathsf{SAT}, \mathsf{SAT})$; see [28] for a more extensive discussion of this topic). Therefore, $\mathsf{UniqueSAT}$ is $\leq_m^{\mathcal{RP}}$-hard for $\mathcal{NP}$.

In [2, 3, 42] some problems are shown to be $\leq_T^{co\mathcal{RP}}$-complete (in fact, $\leq_m^{co\mathcal{RP}}$-complete) for $\mathcal{NP}$. These problems are not known to be $\mathcal{NP}$-complete in the usual sense.

Furthermore, in [38] a tally language is shown to be $\leq_T^{co\mathcal{RP}}$-complete (in fact, $\leq_m^{co\mathcal{RP}}$-complete) for all sparse sets in $\mathcal{NP}$.

The above definitions are asymmetric in the sense that the reduction machine is only allowed to have one-sided errors (with not too large probability). We can also consider two different kinds of symmetric versions of these definitions (corresponding to the so-called "Monte Carlo" and "Las Vegas" types of random computations).

Definition 2.5

a) A set A is polynomial-time randomly Turing reducible with bounded-error *to a set B, denoted by $A \leq_T^{\mathcal{BPP}} B$, if there is a polynomial-time deterministic oracle Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,*

$$M^B(x,r) = [x \in A] \textit{ with probability } \geq 1/2 + 1/q(n).$$

(Again, the string r is drawn uniformly at random from $\{0,1\}^{p(n)}$, for some polynomial p). For some proposition P, we use the notation

$$[P] = \begin{cases} 1, & P \textit{ is true,} \\ 0, & P \textit{ is false.} \end{cases}$$

b) *A set A is polynomial-time* randomly Turing reducible with zero-error *to a set B, denoted by $A \leq_T^{\mathcal{ZPP}} B$, if there is a polynomial-time deterministic oracle Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,*

$$\begin{array}{l} M^B(x,r) = [x \in A] \text{ with probability } \geq 1/q(n), \\ M^B(x,r) = [x \notin A] \text{ with probability } = 0. \end{array}$$

Here, the machine M is allowed to output 0,1, or ? ("don't know").

Instead of allowing a $\mathcal{ZPP}$-reduction machine a ternary output from $\{0, 1, ?\}$ we can obtain an equivalent definition of $\leq_T^{\mathcal{ZPP}}$ by insisting on a definite (and correct) binary output from $\{0, 1\}$, but at the same time, we have to relax the strict polynomial-time bound (for all inputs (x, r)) to just *expected* polynomial time (i.e. for each x, $|x| = n$, the expectation of M's running time on input (x, r) is polynomially bounded in n where the second input argument is drawn at random.) If moreover, M asks exactly one positive query, we say that A is polynomial-time *randomly many-one reducible with zero-error* to B, denoted by $A \leq_m^{\mathcal{ZPP}} B$. Equivalently, there is a polynomial-time deterministic Turing machine M and a polynomial q such that for all inputs x with $|x| = n$,

- $M(x, r)$ outputs ? with probability at most $1 - 1/q(n)$, and
- if $M(x, r) \neq ?$, then $M(x, r) \in B$ if and only if $x \in A$.

Here, the machine M is allowed to output some string y or ? ("don't know").

It is clear that $A \leq_T^{\mathcal{ZPP}} B$ implies both, $A \leq_T^{\mathcal{RP}} B$ and $A \leq_T^{co\mathcal{RP}} B$, which in turn imply $A \leq_T^{\mathcal{BPP}} B$. (An analogous statement holds for the case of the many-one versions of these reducibilities).

Next we define non-uniform reducibilities.

Definition 2.6 *A set A is polynomial-time* non-uniformly Turing reducible *to a set B, denoted by $A \leq_T^{\mathcal{P}/poly} B$, if there is a polynomial-time deterministic oracle Turing machine M and a polynomial q such that for all n there is a string $w \in \{0, 1\}^{q(n)}$ (the "advice" string) such that for all inputs x with $|x| = n$,*

$$M^B(x, w) = [x \in A].$$

The above definition using advice strings is adopted from [21]. It is not hard to see that $A \leq_T^{\mathcal{P}/poly} B$ is equivalent to the existence of a (non-uniform) family of polynomial-size circuits such that the nth circuit has n inputs and recognizes A on strings of length n (see, for example, [28]). Furthermore, such circuits are equipped with oracle gates whose (binary) output depends on the oracle language B. More precisely, an oracle gate (with an arbitrary number of inputs) outputs 1 if and only if the 0-1-string at its inputs is a member of B.

We call such circuits *oracle circuits* in the following. (This relativized circuit model has been introduced by Wilson [43]).

Using an argument given in [8] (see also [1]) it can be seen that $A \leq_T^{\mathcal{BPP}} B$ implies $A \leq_T^{\mathcal{P}/poly} B$.

Definition 2.7 *Let $\leq^{\mathcal{C}}$ be any of the reducibilities defined above. If a set A has the property that for every $L \in \mathcal{NP}$ it holds $L \leq^{\mathcal{C}} A$, then A is called $\leq^{\mathcal{C}}$*-hard *for $\mathcal{NP}$. If additionally $A \in \mathcal{NP}$ holds, then A is called $\leq^{\mathcal{C}}$*-complete *for $\mathcal{NP}$.*

From each of the reducibilities of the type $\leq_T^{\mathcal{C}}$ as defined above, we can obtain a complexity class, denoted $\mathcal{C}$, by letting

$$\mathcal{C} = \{A \mid A \leq_T^{\mathcal{C}} \emptyset\}.$$

This way, we can define the classes $\mathcal{P}$, $\mathcal{SNP}$ (which is the same as $\mathcal{NP} \cap co\mathcal{NP}$), $\mathcal{RP}$, $co\mathcal{RP}$, $\mathcal{ZPP}$, $\mathcal{BPP}$, and $\mathcal{P}/poly$.

Instead of $A \leq_T^{\mathcal{C}} B$, we write sometimes $A \in \mathcal{C}(B)$ where $\mathcal{C}$ is from $\{\mathcal{P}, \mathcal{NP}, co\mathcal{NP}, \mathcal{SNP}, \mathcal{RP}, co\mathcal{RP}, \mathcal{ZPP}, \mathcal{BPP}, \mathcal{P}/poly\}$. We do this whenever our point of view is not the reducibility concept, i.e. a comparison of the relative complexity of A and B, but the operator aspect, i.e. $\mathcal{C}$ maps languages (or classes of languages) to classes of languages.

Our main contribution is a proof that every $\leq_T^{\mathcal{P}/poly}$-hard set A for $\mathcal{NP}$ is $\mathcal{ZPP}(\mathcal{NP})$-high. This means, that any $\mathcal{ZPP}(\mathcal{NP}(\mathcal{NP}))$ computation can be replaced by a $\mathcal{ZPP}(\mathcal{NP}(A))$ computation, as indicated by the following figure.

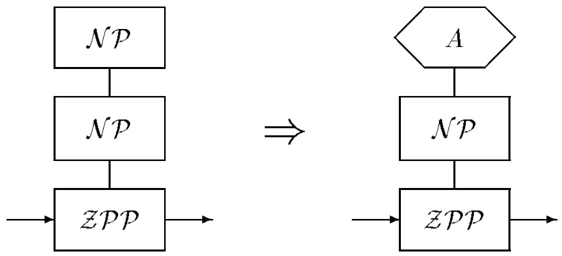

Figure 2: A $\mathcal{ZPP}(\mathcal{NP})$-high set A.

At the core of our proof we exploit the hardness of A to design a randomized algorithm T which, on input 0^m, computes an oracle circuit $\hat{c}$ that decides with oracle A the satisfiability of all formulas of size m. Actually, our proof goes through for any $\mathcal{NP}$ set having *self-computable witnesses* (in the sense of [10, 6, 36, 28]). Related proofs can be found in [11, 29] (the proof given in [29] works for any self-reducible set).

Figure 3 on the next page shows the known inclusions between various classes of high and hard sets for $\mathcal{NP}$. The inclusions indicated by a line directly follow from the definitions or have been explained above. Also, we have already argued that the $\mathcal{NP}$-high sets are exactly the sets that are $\leq_T^{\mathcal{SNP}}$-hard for $\mathcal{NP}$. In the next two lemmas we prove the remaining equivalences.

Lemma 2.8 *For any set A, the following conditions are equivalent:*

(i) *A is $\mathcal{BPP}$-high,*

(ii) *A is $\leq_T^{\mathcal{BPP}}$-hard for $\mathcal{NP}$,*

(iii) *A is $\leq_T^{\mathcal{RP}}$-hard for $\mathcal{NP}$.*

Proof. The implication (i) $\Rightarrow$ (ii) is obvious. Condition (ii) implies that SAT belongs to $\mathcal{BPP}(A)$. Hence, SAT belongs to $\mathcal{RP}(A)$ [23, 45] (see also [35, 28]), which implies (iii). The implication (iii) $\Rightarrow$ (i) is a consequence of the fact that $\mathcal{BPP}$ (and hence $\mathcal{RP}$) is low for $\mathcal{BPP}$ [23, 44]: if $\mathcal{NP} \subseteq \mathcal{RP}(A)$, then $\mathcal{BPP}(\mathcal{NP}) \subseteq \mathcal{BPP}(\mathcal{RP}(A)) = \mathcal{BPP}(A)$. □

Lemma 2.9 *For any set A, the following conditions are equivalent:*

(i) *A is $\mathcal{RP}$-high,*

(ii) *A is $\mathcal{ZPP}$-high,*

(iii) *A is $\leq_T^{\mathcal{ZPP}}$-hard for $\mathcal{NP}$,*

(iv) *A is $\leq_T^{co\mathcal{RP}}$-hard for $\mathcal{NP}$.*

Proof. To see the implication (i) $\Rightarrow$ (ii), assume that $\mathcal{RP}(\mathsf{SAT}) \subseteq \mathcal{RP}(A)$. This implies that SAT (and hence $\mathcal{NP}$) belongs to $\mathcal{ZPP}(A)$, and since $\mathcal{ZPP}$ is low for $\mathcal{ZPP}$ [44], we have $\mathcal{ZPP}(\mathcal{NP}) \subseteq \mathcal{ZPP}(\mathcal{ZPP}(A)) = \mathcal{ZPP}(A)$.

The implications (ii) $\Rightarrow$ (iii) $\Rightarrow$ (iv) are obvious. Since (iv) implies that SAT belongs to $\mathcal{ZPP}(A)$, we get $\mathcal{RP}(\mathsf{SAT}) \subseteq \mathcal{RP}(\mathcal{ZPP}(A)) = \mathcal{RP}(A)$. □

3 Universal Hashing

In our proof in the next section we make use of the universal hashing technique, introduced by [12].

Definition 3.1 *We call a class $\mathcal{H}$ of hash functions mapping elements from A to elements in B* universal*, if for any pair of distinct elements $x_1, x_2 \in A$,*

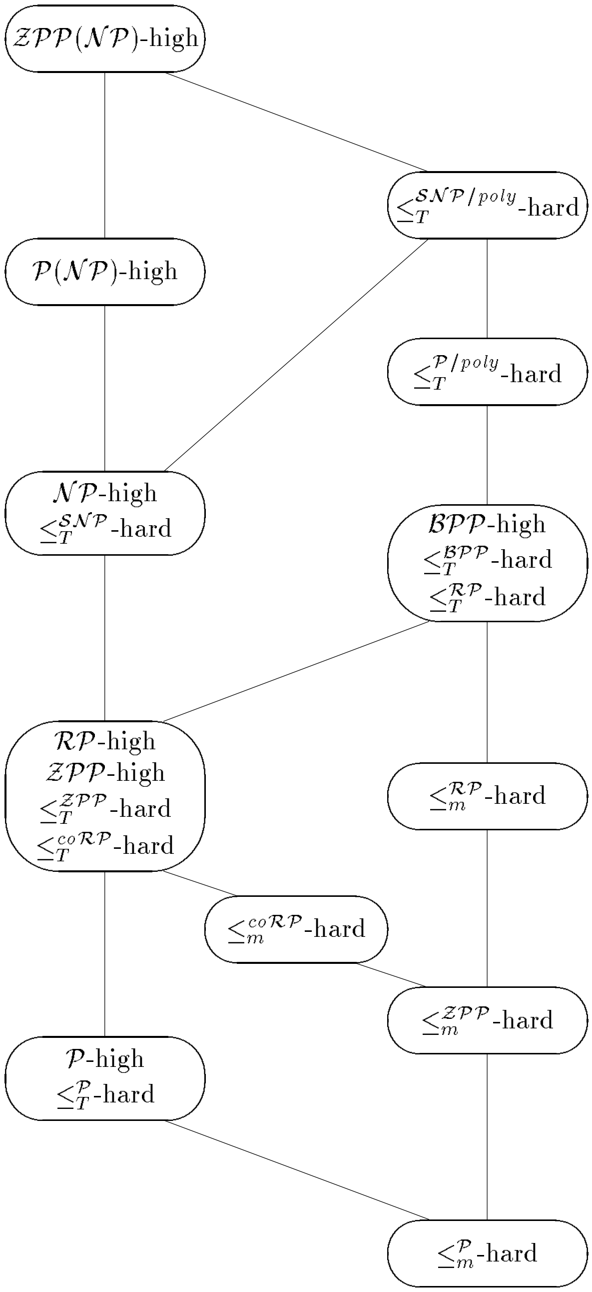

Figure 3: The inclusion structure of high sets and hard sets for $\mathcal{NP}$.

the two random variables $h(x_1)$ and $h(x_2)$ obtained by drawing h under uniform distribution from $\mathcal{H}$ are independently and uniformly distributed over B. This means that for any pair of (not necessarily distinct) elements $y_1, y_2 \in B$,

$$\mathbf{Pr}[h(x_1) = y_1, h(x_2) = y_2] \quad = \quad \frac{1}{||B||^2}.$$

We will make use of the hash class containing all linear functions from Σ^m to Σ^k that we denote by $\mathcal{L}(m,k)$. As usual, we interpret Σ^m and Σ^k as vector spaces over GF[2]. Since any linear hash function h from Σ^m to Σ^k is represented by a Boolean $(m \times k)$-matrix, we can randomly choose a linear hash function from $\mathcal{L}(m,k)$ by generating a sequence of $m \cdot k$ independent random bits and composing them to an $(m \times k)$ matrix. In this way, any linear hash function $h \in \mathcal{L}(m,k)$ is obtained with uniform probability 2^{-mk}. We use the following lemma from [16] to derive the universality of $\mathcal{L}(m,k)$.

Lemma 3.2 *Let $x_1, \ldots, x_l$ be any collection of linearly independent vectors from Σ^m. Then for a uniformly chosen linear hash function h from $\mathcal{L}(m,k)$, the values $h(x_1), \ldots, h(x_l)$ are independently and uniformly distributed over Σ^k.*

Proof. Since the vectors $x_1, \ldots, x_l$ are linearly independent, they can be extended to a basis $x_1, \ldots, x_l, x_{l+1}, \ldots, x_m$ of Σ^m. Let T be the linear transformation that maps this basis to the canonical basis of Σ^m. For any hash function $h \in \mathcal{L}(m,k)$, we denote by h_T the composition of h with T. Since the mapping $h \mapsto h_T$ is a bijection on $\mathcal{L}(m,k)$, it follows that

$$\begin{aligned}
\mathbf{Pr}[h(x_1) = y_1, \ldots, h(x_l) = y_l] &= \mathbf{Pr}[h_T(x_1) = y_1, \ldots, h_T(x_l) = y_l] \\
&= \mathbf{Pr}[h(e_1) = y_1, \ldots, h(e_l) = y_l] \\
&= 2^{-kl}.
\end{aligned}$$

Here, $e_i = 0^{i-1}10^{m-i-1}$, $1 \leq i \leq m$, are the unit vectors of Σ^m, and h is chosen uniformly at random from $\mathcal{L}(m,k)$. The last equality follows from the fact that $h(e_i)$ is just the ith row of the matrix that corresponds to h. $\square$

Since any two distinct vectors $x_1, x_2 \in \Sigma^m - \{0^m\}$ are linearly independent, it follows that for any pair of integers $m, k \geq 1$, the class $\mathcal{L}(m,k)$ is universal on the domain $\Sigma^m - \{0^m\}$.

The hashing technique was first applied to complexity theory by Sipser [40]. The next lemma provides a lower bound on the probability that a randomly chosen hash function h from $\mathcal{L}(m,k)$ *isolates* some element x within a given set S of appropriate size (meaning that x is the only element in S such that $h(x) = 0^k$). Furthermore, it provides an upper bound on the probability that such an x lies in a given small subset S' of S.

Lemma 3.3 *Let $S \subseteq \Sigma^m - \{0^m\}$ be a non-empty set of cardinality s, let $S' \subseteq S$ be of cardinality at most $s/6$, and let k be an integer such that $2^k < 3s \leq 2^{k+1}$. Then, for a uniformly chosen hash function h from $\mathcal{L}(m,k)$,*

(i) *with probability at least $2/9$, there exists exactly one element $x \in S$ such that $h(x) = 0^k$, and*

(ii) *with probability at most $1/9$, there exists some element $x \in S'$ such that $h(x) = 0^k$.*

Proof (following Gupta [18]). We want to show that $s_h = 1$ with probability at least $2/9$, where s_h denotes the cardinality of the set $\{x \in S \mid h(x) = 0^k\}$. By the Inclusion-Exclusion Principle and using the fact that $\mathcal{L}(m,k)$ is a universal class, it follows that

$$\begin{aligned} \mathbf{Pr}[\, s_h = 0\,] &= 1 - \mathbf{Pr}[\,\exists x \in S : h(x) = 0^k\,] \\ &\leq 1 - \sum_{x \in S} \mathbf{Pr}[\,h(x) = 0^k\,] + \sum_{\substack{x,x' \in S,\\ x < x'}} \mathbf{Pr}[\,h(x) = h(x') = 0^k\,] \\ &= 1 - s2^{-k} + \tbinom{s}{2} 2^{-2k}. \end{aligned}$$

On the other hand,

$$\mathbf{Pr}[\, s_h \geq 2\,] \;\leq\; \sum_{\substack{x,x' \in S,\\ x < x'}} \mathbf{Pr}[\,h(x) = h(x') = 0^k\,] \;=\; \tbinom{s}{2} 2^{-2k}.$$

Putting these inequalities together, we get

$$\begin{aligned} \mathbf{Pr}[\, s_h = 1\,] &= 1 - \mathbf{Pr}[\, s_h = 0\,] - \mathbf{Pr}[\, s_h \geq 2\,] \\ &\geq 1 - (1 - s2^{-k} + \tbinom{s}{2} 2^{-2k}) - \tbinom{s}{2} 2^{-2k} \\ &= s2^{-k} - 2\tbinom{s}{2} 2^{-2k} \\ &> s2^{-k}(1 - s2^{-k}) \;\geq\; 2/9, \text{ since } s2^{-k} \in [\tfrac{1}{3}, \tfrac{2}{3}]. \end{aligned}$$

This proves (i). Furthermore, since the cardinality of S' is at most $s/6 \leq 2^{k+1}/18$, the probability that h maps some element from S' to 0^k is at most $(2^{k+1}/18) \cdot 2^{-k} = 1/9$. □

4 Highness of P/POLY Hard Sets

In this section we prove that all $\mathcal{P}/poly$-hard sets for $\mathcal{NP}$ are $\mathcal{ZPP}(\mathcal{NP})$-high. As preparation to the proof, we observe (as in [19, 5]) that any (non-uniform) family of circuits for the $\mathcal{NP}$-complete set SAT can be converted into a new (non-uniform) circuit family in which the circuits are still polynomial in their input size, and not only output a binary value depending on whether the input formula F is satisfiable, but also output a "witness" for satisfiability, i.e. a satisfying assignment (if one exists). Such witness-constructing circuits can be obtained via the self-reducibility of SAT by building a cascade of several original circuits, as illustrated in Figure 4. The triangles indicate original circuits with

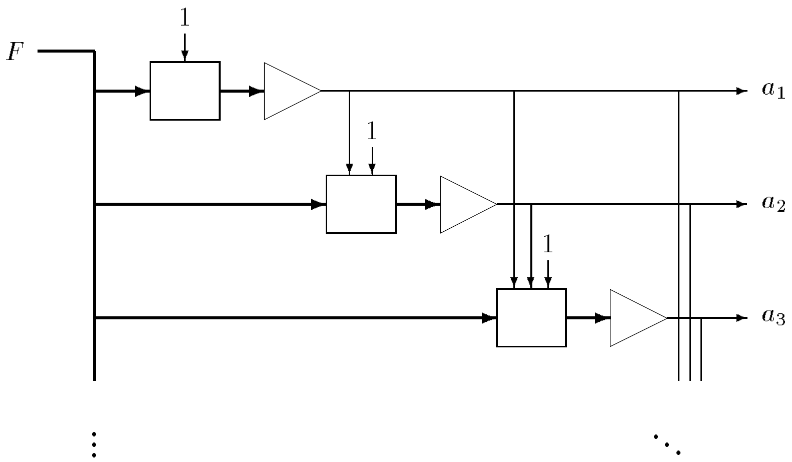

Figure 4: A witness-constructing circuit for SAT.

binary output, whereas a box indicates a circuit that transforms (the binary encoding of) $F = F(x_1, \ldots, x_n)$, where the x_i are Boolean variables, into (the encoding of) $F(a_1, \ldots, a_k, x_{k+1}, \ldots, x_n)$. The binary values $a_1, \ldots, a_k$ are given by the top inputs. Bold lines indicate that a bunch of bits (encoding F or formulas of the form $F(a_1, \ldots, a_k, x_{k+1}, \ldots, x_n)$) are transmitted. Of course, the same construction can be performed for circuits that have access to some oracle A.

Now we are ready to prove our highness result.

Theorem 4.1 *Any $\leq_T^{\mathcal{P}/poly}$-hard set for $\mathcal{NP}$ is $\mathcal{ZPP}(\mathcal{NP})$-high.*

Proof. Let A be a $\leq_T^{\mathcal{P}/poly}$-hard set for $\mathcal{NP}$. We have to show that every $\mathcal{ZPP}(\mathcal{NP}(\mathsf{SAT}))$ computation can be replaced by a $\mathcal{ZPP}(\mathcal{NP}(A))$ computation (as indicated by Figure 2). We can assume that any 3-CNF formula over m variables is encoded by a string x of length $q(m)$, for some polynomial q. We denote the set $\mathsf{SAT} \cap \Sigma^{q(m)}$ of all (encodings of) satisfiable 3-CNF formulas over m variables by SAT_m. By assumption, SAT belongs to $\mathcal{P}^A/poly$, and by the discussion above, there exists a sequence of poly-size oracle circuits c_m with $q(m)$ input gates and m output gates such that for all $x \in \Sigma^{q(m)}$,

$$x \in \mathsf{SAT} \Leftrightarrow c_m(x, A) \text{ outputs a satisfying assignment for } x.$$

Later in the proof we will design a probabilistic algorithm T that runs in expected polynomial time, and that on input 0^m outputs some oracle circuit $\hat{c}$ such that for all $x \in \Sigma^{q(m)}$,

$$\hat{c}(x, A) = 1 \Leftrightarrow x \in \mathsf{SAT}.$$

Using this algorithm, it is not hard to derive the $\mathcal{ZPP}(\mathcal{NP})$-highness of A. Since $\mathcal{ZPP}(\mathcal{ZPP}) \subseteq \mathcal{ZPP}$ via a proof that relatives [44], it suffices to show $\mathcal{NP}(\mathsf{SAT}) \subseteq \mathcal{ZPP}(\mathcal{NP}(A))$. Let $L \in \mathcal{NP}(\mathsf{SAT})$ via some polynomial-time nondeterministic oracle Turing machine M. We may assume that all oracle queries of M on inputs of length n have length $l(n)$, for some fixed polynomial l. Consider the $\mathcal{NP}(A)$ oracle set B defined as

$$\langle x, \hat{c}\rangle \in B \quad \Leftrightarrow \quad \begin{array}{l} M(x) \text{ accepts where each oracle query} \\ \text{`}y \in \mathsf{SAT}\text{' is replaced by `}\hat{c}(y, A) = 1\text{'.} \end{array}$$

Then $L \in \mathcal{ZPP}(\mathcal{NP}(A))$ via the following algorithm:

input x ($|x| = n$);
run T on input $0^{l(n)}$ and obtain a circuit $\hat{c}$ (T uses an $\mathcal{NP}(A)$ oracle)
if $\langle x, \hat{c}\rangle \in B$ **then accept else reject end**

Our next goal is to describe and analyze the probabilistic algorithm T. Without limitation of generality, let the length of c_m (encoded in binary) be $p(m)$, for some fixed polynomial p. We denote the set of all oracle circuits of length $p(m)$ having $q(m)$ input gates and m output gates by C_m. In fact, $\hat{c}$ will be constructed by composing a polynomial number of circuits from C_m such that any $x \in \mathsf{SAT}_m$ is accepted by at least one of these circuits. More formally, for any subset C of C_m, $\mathrm{OR}(\mathsf{C})$ denotes the circuit $\hat{c}$ obtained by combining the circuits in C in such a way that for all $x \in \Sigma^{q(m)}$,

$$\hat{c}(x, A) = 1 \;\Leftrightarrow\; \exists c \in \mathsf{C} : c(x, A) \text{ outputs a satisfying assignment for } x.$$

Note that no matter how C looks like, $\mathrm{OR}(\mathsf{C})$ always rejects all inputs $x \notin \mathsf{SAT}_m$. To describe our algorithm we need one more notation: For a set $S \subseteq \mathsf{SAT}_m$, let $\mathsf{C}^+(S, A)$ denote the subset of C_m consisting of all circuits c that are correct on S, i.e., under oracle A, c computes a satisfying assignment for every $x \in S$. Observe that, by assumption, $\mathsf{C}^+(S, A)$ always contains c_m.

On input 0^m, the following algorithm T iteratively collects satisfiable formulas x into the set S, until it finds a circuit $\hat{c}$ for SAT_m. The goal in expanding

S is to reduce the number of circuits in $\mathsf{C}^+(S,A)$. Hence, T tries to expand S only by formulas x which have the property that $\mathsf{C}^+(S \cup \{x\}, A)$ contains significantly less circuits than $\mathsf{C}^+(S,A)$. To achieve this, T randomly chooses a collection of $9q(m)$ many hash functions h_i and computes the set C of (at most $9q(m)$ many) circuits that are isolated within $\mathsf{C}^+(S,A)$ by some h_i. Then T selects some (say, the lexicographically least) formula $x \in \mathsf{SAT}_m$ such that none of the circuits in C is able to compute a satisfying assignment for x.

> **input** 0^m;
> $S := \emptyset$;
> **loop**
>   **choose randomly** $k \in \{1, \ldots, p(m)+1\}$;
>   **choose randomly** $h_1, \ldots, h_{9q(m)}$ from $\mathcal{L}(p(m), k)$;
> $\hat{c} := \mathrm{OR}\{c \mid \text{some } h_i \text{ isolates } c \text{ within } \mathsf{C}^+(S,A)\}$
>   **if** $\hat{c}(x,A) = 0$ for some $x \in \mathsf{SAT}_m$ **then** $S := S \cup \{x\}$
>   **else exit(loop) end**
> **end loop**;
> **output** $\hat{c}$

It is easy to see that T can be implemented by asking an oracle from $\mathcal{NP}(A)$. Moreover, since $\hat{c}$ never accepts an input $x \notin \mathsf{SAT}_m$ and since the loop only terminates when $\hat{c}$ accepts all $x \in \mathsf{SAT}_m$, the algorithm is obviously correct. It thus remains to show that the expected running time of T is polynomially bounded.

We say that a formula x provides a *successful* extension of S if the size of $\mathsf{C}^+(S,A)$ decreases by a factor smaller than $5/6$, i.e.

$$\|\mathsf{C}^+(S \cup \{x\}, A)\| \quad < \quad (5/6) \cdot \|\mathsf{C}^+(S,A)\|.$$

By the following claim, the cardinality of $\mathsf{C}^+(S,A)$ decreases very rapidly with high probability.

Claim. *Each single extension of S is successful with probability $\geq 1/2p(m)$.*

Note that at the beginning, $\mathsf{C}^+(S,A)$ contains at most $2^{p(m)}$ elements. Since $\mathsf{C}^+(S,A)$ cannot become empty, there are at most $p(m)\log^{-1}(6/5) < 4p(m)$ successful extensions of S. Hence, using the claim, it follows that the expected number of loop iterations is at most $8p^2(m)$.

To prove the claim we make use of the following subclaim.

Subclaim. *Under the condition that $2^k < 3\|\mathsf{C}^+(S,A)\| \leq 2^{k+1}$, a single extension of S is successful with probability $\geq 1-(2/e)^{q(m)}$.*

Since the probability for the condition in the subclaim is exactly $\frac{1}{p(m)+1}$, it follows that the (unconditional) probability for a successful extension is at least

$$\frac{1}{p(m)+1}\left(1-(2/e)^{q(m)}\right).$$

Since this bound is larger than $1/2p(m)$ for sufficiently large m, the claim follows.

It remains to prove the subclaim. Since S is only expanded by formulas $x \in \mathsf{SAT}_m$ with $\hat{c}(x, A) = 0$, it suffices to bound the probability that $\hat{c}(x, A) = 0$ holds for some formula x which does not provide a successful extension of S. Consider a fixed "unsuccessful" $x \in \mathsf{SAT}_m$, i.e., $\mathsf{C}^+(S, A) - \mathsf{C}^+(S \cup \{x\}, A)$ contains at most $\|\mathsf{C}^+(S, A)\|/6$ circuits. Note that $\hat{c}(x, A) = 0$ if and only if none of the hash functions h_i is able to isolate within $\mathsf{C}^+(S, A)$ a circuit $c \in \mathsf{C}^+(S \cup \{x\}, A)$. Now, it follows

- by part (i) of Lemma 3.3 that a single h_i isolates some circuit within $\mathsf{C}^+(S, A)$ with probability greater than $2/9$, and
- by part (ii) of Lemma 3.3 that h_i isolates within $\mathsf{C}^+(S, A)$ a circuit $c \notin \mathsf{C}^+(S \cup \{x\}, A)$ with probability at most $1/9$.

Hence, with probability at least $1/9$, a single h_i isolates within $\mathsf{C}^+(S, A)$ a circuit $c \in \mathsf{C}^+(S \cup \{x\}, A)$. Thus, the probability that none of $h_1, \ldots, h_{9q(m)}$ is able to isolate within $\mathsf{C}^+(S, A)$ a circuit $c \in \mathsf{C}^+(S \cup \{x\}, A)$ is at most $(8/9)^{9q(m)} < e^{-q(m)}$. Since SAT_m contains no more than $2^{q(m)}$ formulas, this gives the desired bound $(2/e)^{q(m)}$. $\square$

We notice that one can easily extend the proof to show that even all $\leq_T^{\mathcal{SNP}/poly}$-hard sets for $\mathcal{NP}$ are $\mathcal{ZPP}(\mathcal{NP})$-high (see Figure 3). Also, another generalization in terms of extended highness (cf. [7]) is easily obtained; i.e. $\mathcal{ZPP}(\mathcal{NP}(\mathsf{SAT} \oplus A)) \subseteq \mathcal{ZPP}(\mathcal{NP}(A))$.

References

[1] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on the Foundations of Computer Science*, 75–83. IEEE Computer Society Press, 1978.

[2] L. Adleman and K. Manders. Reducibility, randomness, and intractibility. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, 151–163. ACM Press, 1977.

[3] L. Adleman and K. Manders. Reductions that lie. In *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science*, 397–410. IEEE Computer Society Press, 1979.

[4] E. Allender and L. Hemachandra. Lower bounds for the low hierarchy. *Journal of the ACM*, **39**:234–250, 1992.

[5] V. Arvind, J. Köbler, U. Schöning, and R. Schuler. If NP has polynomial-size circuits, then MA=AM. *Theoretical Computer Science*, **137**(2):279–282, 1995.

[6] J. Balcázar. Self-reducibility structures and solutions of NP problems. In *Revista Matematica*, 175–184. Universidad Complutense de Madrid, 1989.

[7] J. Balcázar, R. Book, and U. Schöning. Sparse sets, lowness and highness. *SIAM Journal on Computing*, **23**:679–688, 1986.

[8] C.H. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq co\text{-}NP^A$ with probability 1. *SIAM Journal on Computing*, **10**:96–113, 1981.

[9] R. Book, P. Orponen, D. Russo, and O. Watanabe. Lowness properties of sets in the exponential-time hierarchy. *SIAM Journal on Computing*, **17**(3):504–516, 1988.

[10] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report 76-284, Dept. Computer Science, Cornell University, 1976.

[11] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. Technical Report TR 95-015, ECCC, 1995. To appear in *Journal of Computer and System Sciences*.

[12] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, **18**:143–154, 1979.

[13] R. Chang, J. Kadin, and P. Rohatgi. Connections between the complexity of unique satisfiability and the threshold behavior of randomized reductions. In *Proceedings of the 6th Structure in Complexity Theory Conference*, 255–269. IEEE Computer Society Press, 1991.

[14] M. J. Chung and B. Ravikumar. Strong nondeterministic Turing reduction–A technique for proving intractability. *Journal of Computer and System Sciences*, **39**(1):2–20, 1989.

[15] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, 151–158. ACM Press, 1971.

[16] L. Fortnow. *Complexity-Theoretic Aspects of Interactive Proof Systems*. PhD thesis, MIT, 1989.

[17] M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman and Company, 1979.

[18] S. Gupta. E-mail message on the TheoryNet, December 1995.

[19] J. E. Hopcroft. Recent directions in algorithmic research. In P. Deussen, editor, *Proceedings of the 5th Conference on Theoretical Computer Science*, Lecture Notes in Computer Science #104, 123–134. Springer-Verlag, 1981.

[20] J. Kämper. Non-uniform proof systems: A new framework to describe non-uniform and probabilistic complexity classes. *Theoretical Computer Science*, **85**(2):305–331, 1991.

[21] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, 302–309. ACM Press, 1980.

[22] A. Klapper. Generalized lowness and highness and probabilistic complexity classes. *Mathematical Systems Theory*, **22**:37–45, 1989.

[23] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Information Processing Letters*, **14**:39–43, 1982.

[24] K. Ko and U. Schöning. On circuit-size complexity and the low hierarchy in NP. *SIAM Journal on Computing*, **14**:41–51, 1985.

[25] J. Köbler. Locating P/poly optimally in the extended low hierarchy. *Theoretical Computer Science*, **134**(2):263–285, 1994.

[26] J. Köbler. On the structure of low sets. In *Proceedings of the 10th Structure in Complexity Theory Conference*, 246–261. IEEE Computer Society Press, 1995.

[27] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, **44**(2):272–286, 1992.

[28] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhäuser, Boston, 1993.

[29] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. In *Proceedings of the 22nd International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science #944, 196–207. Springer-Verlag, 1995.

[30] T. Long. Strong nondeterministic polynomial-time reducibilities. *Theoretical Computer Science*, **21**:1–25, 1982.

[31] T. Long and M. Sheu. A refinement of the low and high hierarchies. Technical Report OSU-CISRC-2/91-TR6, The Ohio State University, 1991.

[32] D. A. Plaisted. New NP-hard and NP-complete polynomial and integer divisibility problems. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science*, 241–253. IEEE Computer Society Press, 1977.

[33] D. A. Plaisted. Complete divisibility problems for slowly utilized oracles. *Theoretical Computer Science*, **35**:245–260, 1985.

[34] U. Schöning. A low and a high hierarchy within NP. *Journal of Computer and System Sciences*, **27**:14–28, 1983.

[35] U. Schöning. *Complexity and Structure*, Lecture Notes in Computer Science #211. Springer-Verlag, 1986.

[36] U. Schöning. Robust oracle machines. In *Proceedings of the 13th Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science #324, 93–106. Springer-Verlag, 1988.

[37] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, **39**:84–100, 1989.

[38] U. Schöning. On random reductions from sparse sets to tally sets. *Information Processing Letters*, **46**:239–241, 1993.

[39] A. Selman. Polynomial time enumeration reducibility. *SIAM Journal on Computing*, **7**:440–447, 1978.

[40] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, 330–335. ACM Press, 1983.

[41] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, **47**:85–93, 1986.

[42] U. Vazirani and V. Vazirani. A natural encoding scheme proved probabilistic polynomial complete. *Theoretical Computer Science*, **24**:291–300, 1983.

[43] C. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, **31**(2):169–181, 1985.

[44] S. Zachos. Robustness of probabilistic computational complexity classes under definitional perturbations. *Information and Control*, **54**:143–154, 1982.

[45] S. Zachos. Probabilistic quantifiers, adversaries, and complexity classes: an overview. In *Proceedings of the 1st Structure in Complexity Theory Conference*, Lecture Notes in Computer Science #223, 383–400. Springer-Verlag, 1986.