Sparser Random 3-SAT Refutation Algorithms and the Interpolation Problem

Iddo Tzameret*
IIIS, Tsinghua University

Abstract

We formalize a combinatorial principle, called *the 3XOR principle*, due to Feige, Kim and Ofek [14], as a family of unsatisfiable propositional formulas for which refutations of small size in any propositional proof system that possesses the feasible interpolation property imply an efficient *deterministic* refutation algorithm for random 3SAT with n variables and $\Omega(n^{1.4})$ clauses. Such small size refutations would improve the state of the art (with respect to the clause density) efficient refutation algorithm, which works only for $\Omega(n^{1.5})$ many clauses [15].

We demonstrate polynomial-size refutations of the 3XOR principle in resolution operating with disjunctions of quadratic equations with small integer coefficients, denoted R(quad); this is a weak extension of cutting planes with small coefficients. We show that R(quad) is weakly automatizable iff R(lin) is weakly automatizable, where R(lin) is similar to R(quad) but with linear instead of quadratic equations (introduced in [28]). This reduces the problem of refuting random 3CNF with n variables and $\Omega(n^{1.4})$ clauses to the interpolation problem of R(quad) and to the weak automatizability of R(lin).

1 Introduction

In the well known *random 3-SAT model* one usually considers a distribution on formulas in conjunctive normal form (CNF) with m clauses and three literals each, where each clause is chosen independently with repetitions out of all possible $2^3 \cdot \binom{n}{3}$ clauses with n variables (cf. [1]). The *clause density* of such a 3CNF is m/n. When m is greater than cn for sufficiently large c, that is, when the clause density is greater than c, it is known (and easily proved for e.g. $c \geq 5.2$) that with high probability a random 3CNF is unsatisfiable.

A *refutation algorithm* for random kCNFs is an algorithm that receives a kCNF (with c sufficiently large) and outputs either "`unsatisfiable`" or "`don't know`"; if the algorithm answers "`unsatisfiable`" then the kCNF is required to be indeed unsatisfiable; moreover, the algorithm should output "`unsatisfiable`" with high probability (namely, with probability $1 - o(n)$ over the input kCNFs).

We can view the problem of determining the complexity of (deterministic) refutation algorithms as an average-case version of the **P** vs. **coNP** problem: a polynomial-time refutation algorithm for random kCNFs (for a small enough clause density) can be interpreted as showing that "**P** = **coNP** in the average-case"; while a polynomial-time *nondeterministic* refutation algorithm (again, for a small enough clause density) can be interpreted as "**NP** = **coNP** in the average-case".

Refutation algorithms for random kCNFs were investigated in Goerdt and Krivelevich [17] and subsequent works by Goerdt and Lanka [18], Friedman, Goerdt and Krivelevich [16], Feige

*Institute for Theoretical Computer Science, The Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing `tzameret@tsinghua.edu.cn` Supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of P. R. China; Grants 61033001, 61061130540, 61073174, 61373002.

and Ofek [15] and Feige [13] and [10] (among other works). For random 3CNFs, the best (with respect to the clause density) polynomial-time refutation algorithm to date works for formulas with at least $\Omega(n^{1.5})$ clauses [15]. On the other hand, Feige, Kim and Ofek [14] considered efficient *nondeterministic* refutation algorithms; namely, short *witnesses* for unsatisfiability of 3CNFs that can be checked for correctness in polynomial-time. They established the current best (again, with respect to the clause density) efficient, alas *nondeterministic*, refutation procedure: they showed that with probability converging to 1 a random 3CNF with n variables and $\Omega(n^{1.4})$ clauses has a witness of size polynomial in n.

Since the current state of the art random 3CNF refutation algorithm works for $\Omega(n^{1.5})$ clauses, while the best nondeterministic refutation algorithm works already for $O(n^{1.4})$, determining whether a deterministic polynomial-time (or even a quasipolynomial-time) refutation algorithm for random 3CNFs with n variables and $\Omega(n^{1.4})$ clauses exists is to a certain extent the frontier open problem in the area of efficient refutation algorithms.

1.1 Results

In this work we reduce the problem of devising an efficient deterministic refutation algorithm for random 3CNFs with $\Omega(n^{1.4})$ clauses to the interpolation problem in propositional proof complexity. For a refutation system $\mathcal{P}$, the *interpolation problem for* $\mathcal{P}$ is the problem that asks, given a $\mathcal{P}$-refutation of an unsatisfiable formula $A(x,y) \wedge B(x,z)$, for x, y, z mutually disjoint sets of variables, and an assignment α for x, to return 0 or 1, such that if the answer is 0 then $A(\alpha, y)$ is unsatisfiable and if the answer is 1 then $B(\alpha, z)$ is unsatisfiable. If the interpolation problem for a refutation system $\mathcal{P}$ is solvable in time $T(n)$ we say that $\mathcal{P}$ has *interpolation in time* $T(n)$.[1] When $T(n)$ is a polynomial we say that $\mathcal{P}$ has *feasible interpolation*. The notion of feasible interpolation was proposed in [20] and developed further in [30, 7, 22].

We present a family of unsatisfiable propositional formulas, ***denoted*** Υ_n and called *the 3XOR principle formulas*, expressing a combinatorial principle, such that for any given refutation system $\mathcal{P}$ that admits short refutations of Υ_n, solving efficiently the interpolation problem for $\mathcal{P}$ provides an efficient *deterministic* refutation algorithm for random 3CNFs with $\Omega(n^{1.4})$ clauses. In other words, we have the following:

Theorem 1. *If there exists a propositional proof system $\mathcal{P}$ that has interpolation in time $T(n)$ and that admits $s(n)$-size refutations of Υ_n, then there is a deterministic refutation algorithm for random 3CNF formulas with n variables and $\Omega(n^{1.4})$ clauses that runs in time $T(s(n))$. In particular, if $\mathcal{P}$ has feasible interpolation and admits polynomial-size refutations of Υ_n then the refutation algorithm runs in polynomial-time.*

The argument is based on the following: we show that the *computationally hard part* of the Feige, Kim and Ofek nondeterministic refutation algorithm (namely, the part we do not know how to efficiently compute deterministically) corresponds to a disjoint **NP**-pair. Informally, the pair $(\mathbf{A}, \mathbf{B})$ of disjoint **NP** sets is the following: $\mathbf{A}$ is the set of 3CNFs that have a certain combinatorial property, that is, they contain a collection of sufficiently many *inconsistent even k-tuples*, as defined by Feige et al. (see Definition 2); and $\mathbf{B}$ is the set of 3CNFs with m clauses for which there exists an assignment that satisfies more than $m - \ell$ clauses as 3XORs (for ℓ a certain function of the number of variables n).

Theorem 1 then follows from the known relation between disjoint **NP**-pairs and feasible interpolation [29, 27]: in short, if $\mathbf{A}$ and $\mathbf{B}$ are two disjoint **NP** sets and $A(x,y)$ and $B(x,z)$ are the two polynomial-size Boolean formulas corresponding to $\mathbf{A}$ and $\mathbf{B}$, respectively (i.e., for all x, there exists a short y such that $A(x,y) = 1$ iff $x \in \mathbf{A}$; and similarly for $\mathbf{B}$), then short refutations of $A(x,y) \wedge B(x,z)$ imply a polynomial-size algorithm that separates $\mathbf{A}$ from $\mathbf{B}$. For

[1] We do not distinguish in this paper between proofs and refutations: proof systems prove tautologies and refutation systems refute unsatisfiable formulas (or, equivalently prove the negation of unsatisfiable formulas).

more on the relation between disjoint **NP**-pairs and propositional proof complexity see, e.g., [27, 3].

In general, we observe that every efficient refutation algorithm (deterministic or not) corresponds directly to a disjoint **NP**-pair as follows: every efficient refutation algorithm is based on some property P of CNFs that can be witnessed (or better, found) in polynomial-time. Thus, every efficient refutation algorithm corresponds to a family of formulas $P(x) \to \neg\mathrm{SAT}(x)$, expressing that if the input CNF has the property P then x is unsatisfiable; thus, $P(x)$ and $\mathrm{SAT}(x)$ are two disjoint **NP** predicates. In the case of the refutation algorithm of Feige, Kim and Ofek, $P(x)$ expresses simply that the 3CNF x has the Feige et al. witness. *However, the disjoint* **NP***-pair* $(\mathbf{A}, \mathbf{B})$ *we work with is not of this type.* Namely, $\mathbf{A}$ is not the predicate $P(x)$ for the full Feige, Kim and Ofek witnesses, rather a specific combinatorial predicate (mentioned above) that is only one ingredient in the definition of the Feige et al. witness; and $\mathbf{B}$ is not $\mathrm{SAT}(x)$. This saves us the trouble to formalize and prove in a weak propositional proof system the full Feige et al. argument (such a formalization was done recently in [25]; see Sec. 1.2 for a comparison with [25]).

In the second part of this paper (Section 5 onwards) we reduce the problem of determinizing the Feige et al. nondeterministic refutation algorithm to the interpolation problem of a concrete and apparently weak refutation system. Specifically, we demonstrate polynomial-size refutations for Υ_n in a refutation system denoted R(quad) that extends both the cutting planes with small coefficients[2] (cf. [11, 7, 26]) and Res(2) (for any natural k, the system Res(k) is resolution that operates with kDNFs instead of clauses, introduced by Krajíček [23]). We note also that R(quad) is a subsystem of TC^0-Frege.

An R(quad) refutation (see Section 5.1 for a formal definition) over the variables $\{x_1, \ldots, x_n\}$ operates with *disjunctions* of quadratic equations, where each quadratic equation is of the form:

$$\sum_{i,j\in[n]} c_{ij}x_ix_j + \sum_{i\in[n]} c_ix_i + c_0 = a,$$

in which all c_i, c_{ij} and a are integers written in unary representation. The system R(quad) has the following derivation rule, which can be viewed as a generalized resolution rule: from two disjunctions of quadratic equations $\bigvee_i L_i \vee (L = a)$ and $\bigvee_j L_j \vee (L' = b)$ one can derive:

$$\bigvee_i L_i \vee \bigvee_j L_j \vee (L - L' = a - b).$$

We also add axioms that force our variables to be $0, 1$. An R(quad) refutation of an unsatisfiable set of disjunctions of quadratic equations is a sequence of disjunctions of quadratic equations (called *proof-lines*) that terminates with $1 = 0$, and such that every proof-line is either an axiom, or appears in T, or is derived from previous lines by the derivation rules.

We show the following:

Theorem 2. R(quad) *admits polynomial-size refutations of the 3XOR principle formulas* Υ_n.

This polynomial upper bound on the refutation size of the 3XOR principle is non-trivial because the encoding of the 3XOR formula is complicated in itself and further the refutation system is very restrictive.

By Theorem 1, we get the reduction from determinizing Feige et al. work to the interpolation problem for R(quad). In other words:

Corollary 3. *If* R(quad) *has feasible interpolation then there is a deterministic polynomial-time refutation algorithm for random 3CNFs with* n *variables and* $\Omega(n^{1.4})$ *clauses.*

[2] A refutation in *cutting planes with small coefficients* is a restriction of cutting planes in which all intermediate inequalities are required to have coefficients bounded in size by a polynomial in n, where n is the size of the formula to be refuted (see [7]).

Next we reduce the problem of determinizing the Feige et al. refutation algorithm to the weak automatizability of a weaker system than R(quad), namely R(lin), as explained in what follows.

The concept of automatizability, introduced by Bonet, Pitassi and Raz [8] (following the work of [24]), is central to proof-search algorithms. The *proof-search problem* for a refutation system $\mathcal{P}$ asks, given an unsatisfiable formula τ, to find a $\mathcal{P}$-refutation of τ. A refutation system $\mathcal{P}$ is *automatizable* if for any unsatisfiable τ the proof-search problem for $\mathcal{P}$ is solvable in time polynomial in the smallest $\mathcal{P}$-refutation of τ (and equivalently, if there exists a polynomial-time algorithm that on input τ and a number m in unary, outputs a $\mathcal{P}$-refutation of τ of size at most m in, case such a refutation exists). Following Atserias and Bonet [3], we say that a refutation system $\mathcal{P}$ is *weakly automatizable* if there exists an automatizable refutation system $\mathcal{P}'$ that polynomially simulates $\mathcal{P}$. Note that if $\mathcal{P}$ is not automatizable, it does *not* necessarily follow that also $\mathcal{P}'$ is not automatizable. Hence, from the perspective of proof-search algorithms, weak automatizability is a more natural notion than automatizability (see [27] on this).

In [28], the system R(lin) was introduced which is similar to R(quad), except that all equations are *linear* instead of quadratic. In other words, R(lin) is resolution over linear equations with small coefficients. We show the following:

Theorem 4. R(quad) *is weakly automatizable iff* R(lin) *is weakly automatizable.*

The proof of this theorem follows a similar argument to Pudlák [27]. Since weak automatizability of a proof system implies that the proof system has feasible interpolation [8, 27], we obtain the following:

Corollary 5. *If* R(lin) *is weakly automatizable then there is a deterministic refutation algorithm for random 3CNFs with n variables and $\Omega(n^{1.4})$ clauses.*

1.2 Consequences and relations to previous work

The key point of this work is the relation between constructing an efficient refutation algorithm for the clause density $\Omega(n^{0.4})$ to proving upper bounds in weak enough propositional proof systems for the 3XOR principle (namely, proof systems possessing feasible interpolation); as well as establishing such upper bounds in relatively week proof systems.

There are two ways to view our results: either as **(i)** proposing an approach to improve the current state of the art in refutation algorithms via proof complexity upper bounds; or conversely as **(ii)** providing a *new kind* of important computational consequences that will follow from feasible interpolation and weak automatizability of weak proof systems. Indeed, the consequence that we provide is of a different kind from the group of important recently discovered algorithmic-game-theoretic consequences shown by Atserias and Maneva [4], Huang and Pitassi [19] and Beckmann, Pudlák and Thapen [6]. In what follows we explain these two views in more details.

(i) Our results show that by proving that R(quad) has feasible interpolation or by demonstrating a short refutation of the 3XOR principle in some refutation system that admits feasible interpolation, one can advance the state of the art in refutation algorithms. We can hope that if feasible interpolation of R(quad) does not hold, perhaps interpolation in quasipolynomial-time holds (either for R(quad) or for any other system admitting short refutations of the 3XOR principle), which would already improve exponentially the running time of the current best deterministic refutation algorithm for 3CNFs with $\Omega(n^{1.4})$ clauses, since the current algorithm works in time $2^{O(n^{0.2}\log n)}$ [14].

As mentioned above, R(quad) is a common extension of Res(2) and cutting planes with small coefficients (though it is apparently not the weakest such common extension because already R(lin) polynomially simulates both Res(2) and cutting planes with small coefficients). Whether Res(2) and cutting planes with small coefficients have feasible interpolation (let alone,

interpolation in quasi-polynomial time) is open and there are no conclusive evidences for or against it. Note that by Atserias and Bonet [3], Res(2) has feasible interpolation iff resolution is weakly automatizable. However this does not necessarily constitute a strong evidence against the feasible interpolation of Res(2), because the question of whether resolution is *weakly* automatizable is itself open, and there is no strong evidence ruling out a positive answer to this question[3]. Similarly, there are no strong evidences that rule out the possibility that cutting planes is weakly automatizable.

(ii) Even if our suggested approach is not expected to lead to an improvement in refutation algorithms, it is still interesting in the following sense. The fact that R(quad) has short refutations of the 3XOR principle provides a new evidence that (weak extensions of) Res(2) and cutting planes with small coefficients may not have feasible interpolation, or at least that it would be highly non-trivial to prove they do have feasible interpolation; the reason for this is that establishing the feasible interpolation for such proof systems would entail quite strong algorithmic consequences, namely, a highly non-trivial improvement in refutation algorithms. This algorithmic consequence adds to other recently discovered and important algorithmic-game-theoretic consequences that would follow from feasible interpolation of weak proof systems.

Specifically, in recent years several groups of researchers discovered connections between feasible interpolation and weak automatizability of small depth Frege systems to certain game-theoretic algorithms: Atserias and Maneva [4] showed that solving *mean payoff games* is reducible to the weak automatizability of depth-2 Frege (equivalently, Res(n)) systems and to the feasible interpolation of depth-3 Frege systems (actually, depth-3 Frege where the bottom fan-in of formulas is at most two). Subsequently, Huang and Pitassi [19] showed that if depth-3 Frege system is weakly automatizable, then *simple stochastic games* are solvable in polynomial time. Finally, Beckmann, Pudlák and Thapen [6] showed that weak automatizability of resolution implies a polynomial-time algorithm for the *parity game*.

Comparison with Müller and Tzameret [25]. In [25] a polynomial-size TC^0-Frege proof of the correctness of the Feige et al. witnesses was shown. However the goal of [25] was different from the current paper. In [25] the goal was to construct short propositional refutations for random 3CNFs (with sufficiently low clause density). Accordingly, the connection to the interpolation problem was not made in [25]; and further, it is known by [8] that TC^0-Frege does not admit feasible interpolation (under cryptographical assumptions). On the other hand, this paper aims to demonstrate that certain short refutations will have *algorithmic* consequences (for refutation algorithms). Indeed, since we are not interested here to prove the correctness of the full Feige et al. witnesses, we are isolating the computationally hard part of the witnesses from the easy (polytime computable) parts, and formalize the former part (i.e., the 3XOR principle) as a propositional formula in a way that is suitable for the reduction to the interpolation problem.

One advantage of this work over [25] is that Theorem 2 gives a more concrete logical characterization of parts of the Feige et al. witnesses (because the proofs in [25] were conducted indirectly, via a general translation from first-order proofs in bounded arithmetic), and this characterization is possibly *tighter* (because R(quad) is apparently strictly weaker than TC^0-Frege).

2 Preliminaries

Let F be a 3CNF with n variables $X = \{x_1, \ldots, x_n\}$ and m clauses. We denote $\{1, \ldots, n\}$ by $[n]$. The truth value of a formula G under the Boolean assignment A is written $G(A)$. An

[3] It is known that, based on reasonable hardness assumptions from parameterized complexity, resolution is not *automatizable* by Alekhnovich and Razborov [2], which is, as the name indicates, a stronger property than weak automatizability.

assignment A *satisfies as a 3XOR* thew clause $\ell_1 \vee \ell_2 \vee \ell_3$ if $(\ell_1 \oplus \ell_2 \oplus \ell_3)(A) = 1$ (where $\oplus$ denotes the XOR operation, and the ℓ_i's are literals, namely variables or their negation).

2.1 Disjoint NP-pairs and feasible interpolation of propositional proofs

In this section we review the notion of a disjoint **NP**-pair and its relation to propositional proofs and the feasible interpolation property.

A *disjoint **NP**-pair* is simply a pair of languages in **NP** that are disjoint. Let L, N be a disjoint **NP**-pair such that $R(x,y)$ is the corresponding relation for L and $Q(x,z)$ is the corresponding relation for N; namely, there exists polynomials p, q such that $R(x,y)$ and $Q(x,z)$ are polynomial-time relations where $x \in L$ iff $\exists y, |y| \leq p(|x|) \wedge R(x,y) = \texttt{true}$ and $x \in N$ iff $\exists z, |z| \leq q(|x|) \wedge Q(x,z) = \texttt{true}$.

Since both polynomial-time relations $R(x,y)$ and $Q(x,z)$ can be converted into a family of polynomial-size Boolean circuits, they can be written as a family of polynomial-size (in n) CNF formulas (by adding extension variables, that we may assume are incorporated in the certificates y and z). Thus, let $A_n(\overline{x}, \overline{y})$ be a polynomial-size CNF in the variables $\overline{x} = (x_1, \ldots, x_n)$ and $\overline{y} = (y_1, \ldots, y_\ell)$, that is true iff $R(\overline{x}, \overline{y})$ is true, and let $B_n(\overline{x}, \overline{z})$ be a polynomial-size CNF in the variables $\overline{x}$ and $\overline{z} = (z_1, \ldots, z_m)$, that is true iff $Q(\overline{x}, \overline{z})$ is true (for some ℓ, m that are polynomial in n). For every $n \in \mathbb{N}$, we define the following unsatisfiable CNF formula in three mutually disjoint vectors of variables $\overline{x}, \overline{y}, \overline{z}$:

$$F_n := A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z}). \tag{1}$$

Note that because $\overline{y}$ and $\overline{z}$ are disjoint vectors of variables and $A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z})$ is unsatisfiable, it must be that given any $\overline{x} \in \{0,1\}^n$, either $A_n(\overline{x}, \overline{y})$ or $B_n(\overline{x}, \overline{z})$ is unsatisfiable (or both).

Feasible interpolation. We use standard notions from the theory of propositional proof complexity (see [5, 31, 9, 21] for surveys and introductions to the field). In particular, we sometimes mix between refutations (that is, proofs of unsatisfiability of a formula) and proofs (that is, proofs of tautologies). From the perspective of proof complexity refutations of contradictions and proofs of tautologies are for most purposes the same.

A *propositional proof system* $\mathcal{P}$ is a polynomial-time relation $V(\pi, \tau)$ such that for every propositional formulas τ (encoded as binary strings in some natural way), τ is a tautology iff there exists a binary string π (the supposed "proof of τ") with $V(\pi, \tau) = \texttt{true}$. (Note that $|\pi|$ is not necessarily polynomial in $|\tau|$.) A propositional proof system $\mathcal{P}$ *polynomially-simulates* another propositional proof system $\mathcal{Q}$ if there is a polynomial-time computable function f that maps $\mathcal{Q}$-proofs to $\mathcal{P}$-proofs of the same tautologies.

Consider a family of unsatisfiable formulas $F_n := A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z})$, $i \in \mathbb{N}$, in mutually disjoint vectors of variables, as in (1) above. We say that the Boolean function $f(\overline{x})$ is *the interpolant of* F_n if for every n and every assignment $\overline{\alpha}$ to $\overline{x}$:

$$\begin{array}{lcll} f(\overline{\alpha}) = 1 & \Longrightarrow & A_n(\overline{\alpha}, \overline{y}) \text{ is unsatisfiable; and} \\ f(\overline{\alpha}) = 0 & \Longrightarrow & B_n(\overline{\alpha}, \overline{z}) \text{ is unsatisfiable.} \end{array} \tag{2}$$

In other words, if only $A_n(\overline{\alpha}, \overline{y})$ is unsatisfiable (meaning that $B_n(\overline{\alpha}, \overline{z})$ is satisfiable) then $f(\overline{\alpha}) = 1$, and if only $B_n(\overline{\alpha}, \overline{z})$ is unsatisfiable (meaning that $A_n(\overline{\alpha}, \overline{z})$ is satisfiable) then $f(\overline{\alpha}) = 0$, and if both $A_n(\overline{\alpha}, \overline{y})$ and $B_n(\overline{\alpha}, \overline{z})$ are unsatisfiable then $f(\overline{\alpha})$ can be either 0 or 1. Note that L (as defined above) is precisely the set of those assignments $\overline{\alpha}$ for which $A(\overline{\alpha}, \overline{y})$ is satisfiable, and N is precisely the set of those assignments $\overline{\alpha}$ for which $B(\overline{\alpha}, \overline{z})$ is satisfiable, and L and N are disjoint by assumption, and so $f(\overline{x})$ *separates* L from N; namely, it outputs different values for those elements in L and those elements in N.

Definition 1 (Interpolation property)**.** *A propositional proof system $\mathcal{P}$ is said to have the* interpolation property in time $T(n)$ *if the existence of a size $s(n)$ $\mathcal{P}$-refutation of a family F_n as in (1) above implies the existence of an algorithm computing $f(\overline{x})$ in $T(s(n))$ time. When a proof system $\mathcal{P}$ has the interpolation property in time* poly(n) *we say that $\mathcal{P}$ has the* feasible interpolation property*, or simply that $\mathcal{P}$ has* feasible interpolation*.*

2.2 Refutation algorithms

We repeat here the definition given in the introduction. The distribution of *random 3CNF formulas* with n variables and m clauses is defined by choosing m clauses with three literals each, where each clause is chosen independently with repetitions out of all possible $2^3 \cdot \binom{n}{3}$ clauses with n variables. A *refutation algorithm* for random 3CNFs is an algorithm A with input a 3CNF and two possible outputs "`unsatisfiable`" and "`don't know`", such that (i) if on input C, A outputs "`unsatisfiable`", then C is unsatisfiable; and (ii) for any n, with probability at least $1 - o(1)$ A outputs "`unsatisfiable`" (where the probability is considered over the distribution of random 3CNFs with n variables and m clauses, and where $o(1)$ stands for a term that converges to 0 when n tends to infinity).

3 The 3XOR principle

The following definitions and proposition are due to Feige et al. [14].

Definition 2 (Inconsistent even k-tuple)**.** *An* even k-tuple *is a tuple of k many 3-clauses in which every variable appears even times. An* inconsistent even k-tuple *is an even k-tuple in which the total number of negative literals is odd.*

Note that for any even k-tuple, k must be an even number (since by assumption the total number of variables occurrences $3k$ is even). The following is the combinatorial principle, due to Feige et al. [14] that we consider in this work:

The 3XOR Principle. *Let K be a 3CNF over the variables X. Let S be t inconsistent even k-tuples from K, such that every clause from K appears in at most d inconsistent even k-tuples in S. Then, given any Boolean assignment to the variables X, the number of clauses in K that are unsatisfied by the assignment as 3XOR is at least $\lceil t/d \rceil$.*

The correctness of the 3XOR principle follows directly from the following proposition and the fact that every clause in K appears in at most d even k-tuples in S:

Proposition 6 ([14])**.** *For any inconsistent even k-tuple (over the variables X) and any Boolean assignment A to X, there must be a clause in the k-tuple that is unsatisfied as 3XOR.*

The proof follows a simple counting modulo 2. For completeness we prove this proposition.

Proof. Assume by a way of contradiction that for some assignment A every clause from the k-tuple is satisfied as a 3XOR and recall that k must be even. Thus, if we sum modulo 2 all the literals in the k-tuple *via clauses*, then since k is even we get that the sum equals 0 modulo 2.

On the other hand, if we count *via literals* then summing modulo 2 all literals $\ell_i(A)$ in the k-tuple, we get 1 (modulo 2), for the following reason. First, we sum all variables x_i that have odd number of negative occurrences. Because x_i appears an even number of times in the k-tuple, the number of positive occurrences of x_i is also odd. So in total all occurrences of $x_i(A)$ and $\neg x_i(A)$ contribute 1 to our sum (modulo 2). There must be an odd number of such variables x_i in our k-tuple because the k-tuple is *inconsistent*. Thus this sums up to 1 (modulo 2). Then we add to this sum those variables that have an even number of negative occurrences (and hence also an even number of positive occurrences); but they cancel out when summing their values under A modulo 2, and so they contribute 0 to the total sum. Hence, we get 1 as the total sum. This contradicts the counting in the previous paragraph which turned out 0. QED

4 From short proofs to refutation algorithms

In this section we demonstrate that polynomial-size proofs of (encodings of the) 3XOR principle in a proof system that has the feasible interpolation property yield deterministic polynomial-time refutation algorithms for random 3CNF formulas with $\Omega(n^{1.4})$ clauses.

4.1 The witness for unsatisfiability

Feige, Kim and Ofek nondeterministic refutation algorithm [14] is based on the existence of a polynomial-size witness of unsatisfiability for most 3CNF formulas with sufficiently large clause to variable ratio. The witness has several parts, but as already observed in [14], apart from the t inconsistent even k-tuples (Definition 2), all the other parts of the witness are known to be computable in polynomial-time. In what follows we define the witnesses for unsatisfiability.

Let K be a 3CNF with n variables $x_1, \ldots, x_n$ and m clauses. The *imbalance* of a variable x_i is the absolute value of the difference between the number of its positive occurrences and the number of its negative occurrences. The *imbalance of* K is the sum over the imbalances of all variables, in K, denoted $I(K)$. We define $M(K)$ to be an $n \times n$ rational matrix M as follows: let $i, j \in [n]$, and let d be the number of clauses in K where x_i and x_j appear with different signs and s be the number of clauses where x_i and x_j appear with the same sign. Then $M_{ij} := \frac{1}{2}(d - s)$. In other words, for each clause in K in which x_i and x_j appear with the same sign we add $\frac{1}{2}$ to M_{ij} and for each clause in K in which x_i and x_j appear with different signs we subtract $\frac{1}{2}$ from M_{ij}. Let λ be a rational approximation of the biggest eigenvalue of $M(K)$. We shall assume that additive error of the approximation is $1/n^c$ for a constant c independent of n; i.e., $|\lambda - \lambda'| \leq 1/n^c$, for λ' the biggest eigenvalue of $M(K)$; see [25].

Definition 3 (FKO witness)**.** *Given a 3CNF K, the* FKO witness for the unsatisfiability of K *is defined to be the following collection:*

1. *the imbalance $I(K)$;*
2. *the matrix $M(K)$ and the (polynomially small) rational approximation λ of its largest eigenvalue;*
3. *a collection S consisting of $t < n^2$ inconsistent even k-tuples such that every clause in K appears in at most d many even k-tuples, for some positive natural k;*
4. *the inequality $t > \frac{d \cdot (I(K) + \lambda n)}{2} + o(1)$ holds.*

(The $o(1)$ above stands for a specific rational number b/n^c, for c and b constants independent of n).

Feige et al. [14] showed that if a 3CNF has a witness as above it is unsatisfiable. We have the following:

Theorem 7 ([14])**.** *There are constants c_0, c_1 such that for a random 3CNF K with n variables and $\Omega(n^{1.4})$ clauses, with probability converging to 1 as n tends to infinity there exist natural numbers k, t, d such that $t = \Omega(n^{1.4})$ and*

$$k \leq c_0 \cdot n^{0.2} \quad \text{and} \quad t < n^2 \quad \text{and} \quad d \leq c_1 \cdot n^{0.2}, \tag{3}$$

and K has a witness for unsatisfiability as in Definition 3.

Inspecting the argument in [14], it is not hard to see that it is sufficient to replace part 3 in the witness with a witness for the following:

3'. No assignment can satisfy more than $m - \lceil t/d \rceil - 1$ clauses in K as 3XORs.

Therefore, since $I(K)$, $M(K)$ and λ are all polynomial-time computable (see [14] for this), in order to determinize the nondeterministic refutation algorithm of [14] it is sufficient to provide an algorithm that almost surely determines (correctly) that part 3' above holds (when also t and d are such that part 4 in the witness holds). In other words, in order to construct an efficient refutation algorithm for random 3CNFs (with $\Omega(n^{1.4})$ clauses) it is sufficient to have a deterministic algorithm A that on every input 3CNF (and for t and d such that part 4 in the witness holds) answers either "`condition 3' is correct`" or "`don't know`", such that A is never wrong (i.e., if it says "`condition 3' is correct`" then condition 3' holds) and with probability $1-o(n)$ over the input 3CNFs A answers "`condition 3' is correct`". Note that we do not need to actually find the Feige et al. witness nor do we need to decide if it exists or not (it is possible that condition 3' holds but condition 3 does not, meaning that there is *no* Feige et al. witness). The relation between unsatisfiability and bounding the number of clauses that can be satisfied as 3XOR in a 3CNF was introduced by Feige in [12] (and used in [15] as well as in [14]).

4.2 The disjoint NP-pair corresponding to the 3XOR principle

We define the corresponding *3XOR principle disjoint* ***NP****-pair* as the pair of languages (L, N), where k, t, d are natural numbers given in *unary*:

$$L := \{\langle X, k, t, d\rangle \mid X \text{ is a 3CNF with } n \text{ variables and Equation (3) holds for } k, t, d \text{ and there exists } t \text{ inconsistent even } k\text{-tuples such that each clause of } X \text{ appears in no more than } d \text{ many } k\text{-tuples}\},$$

$$N := \{\langle X, k, t, d\rangle \mid X \text{ is a 3CNF with } n \text{ variables and } m \text{ clauses and Equation (3) holds for } k, t, d \text{ and there exists an assignment that satisfies at least } m - \lceil t/d\rceil \text{ clauses in } X \text{ as 3XOR}\}.$$

It is easy to verify that both L and N are indeed **NP** sets, and that by the 3XOR principle, $L \cap N = \emptyset$.

Using the same notation as in Section 2.1, we denote by $R(x, y)$ and $Q(x, z)$ the polynomial-time relations for L and N, respectively. Further, for every $n \in \mathbb{N}$, there exists an *unsatisfiable* CNF formula in three mutually disjoint sets of variables $\overline{x}, \overline{y}, \overline{z}$:

$$\Upsilon_n := A_n(\overline{x}, \overline{y}) \wedge B_n(\overline{x}, \overline{z}), \tag{4}$$

where $A_n(\overline{x}, \overline{y})$ and $B_n(\overline{x}, \overline{z})$ are the CNF formulas expressing that $R(x, y)$ and $Q(x, z)$ are true for x of length n, respectively.

Theorem 1. *Assume that there exists a propositional proof system that has interpolation in time $T(n)$ and that admits size $s(n)$ refutations of Υ_n. Then, there is a deterministic refutation algorithm for random 3CNF formulas with $\Omega(n^{1.4})$ clauses running in time $T(s(n))$.*

Remark 8. *Specifically, if the propositional proof system has feasible interpolation and admits polynomial-size refutations of Υ_n we obtain a polynomial-time refutation algorithm.*

Proof. By the assumption, and by the definition of the feasible interpolation property, there exists a deterministic polynomial-time interpolant algorithm A that on input a 3CNF K and three natural numbers k, t, d given in unary, if $\mathsf{A}(K, k, t, d) = 1$ then $\langle K, k, t, d\rangle \notin L$ and if $\mathsf{A}(K, k, t, d) = 0$ then $\langle K, k, t, d\rangle \notin N$.

The desired refutation algorithm works as follows: it receives the 3CNF K and for each 3-tuple of natural numbers $\langle k, t, d\rangle$ for which Equation (3) holds it runs $\mathsf{A}(K, k, t, d)$. Note there are only $O(n^3)$ such 3-tuples. If for one of these runs $\mathsf{A}(K, k, t, d) = 0$ then we know that

$\langle K, k, t, d\rangle \notin N$; in this case we check (in polynomial-time) that the inequality in Part 4 of the FKO witness (Definition 3) holds, and if it does, we answer "`unsatisfiable`". Otherwise, we answer "`don't know`".

The correctness of this algorithm stems from the following two points:

(i) If we answered "`unsatisfiable`", then there exist k, t, d such that $\langle K, k, t, d\rangle \notin N$ and Part 4 in the FKO witness holds, and so Condition 3' (from Section 4.1) is correct, and hence, by the discussion in 4.1, K is unsatisfiable.

(ii) For almost all 3CNFs we will answer "`unsatisfiable`". This is because almost all of them will have an FKO witness (by Theorem 7), which means that $\langle K, k, t, d\rangle \in L$ for some choice of $t < n^2, d, k$ (in the prescribed ranges) and hence the interpolant algorithm A must output 0 in at least one of these cases (because $\mathsf{A}(K, k, t, d) = 1$ means that $\langle K, k, t, d\rangle \notin L$). `QED`

5 Short refutations of the 3XOR principle

In this section we define the propositional refutation system in which we demonstrate polynomial-size refutations of the 3XOR principle. We then give an explicit encoding of the 3XOR principle as an unsatisfiable set of disjunctions of linear equations.

5.1 The propositional refutation systems R(lin) and R(quad)

The refutation system in which we shall prove the unsatisfiability of the 3XOR principle is denoted R(quad). It is an extension of the refutation system R(lin) introduced in [28]. The system R(lin) operates with disjunctions of linear equations with integer coefficients and R(quad) operates with disjunctions of quadratic equations with integer coefficients, where in both cases the coefficients are written in unary representation. We also add axioms that force all variables to be $0, 1$. First we define the refutation system R(lin).

The **size** of a linear equation $a_1x_1 + ... + a_nx_n + a_{n+1} = a_0$ is defined to be $\sum_{i=0}^{n+1} |a_i|$, that is, the sum of the sizes of all a_i written in *unary* notation. The *size of a disjunction of linear equations* is the total size of all linear equations in it. The **size** of a *quadratic equation* and of a disjunction of quadratic equations is defined in a similar manner (now counting the size of the constant coefficients, the coefficients of the linear terms and the coefficients of the quadratic terms). The *empty disjunction* is unsatisfiable and stands for the truth value false.

Notation: For L a linear or quadratic sum and $S \subseteq \mathbb{Z}$, we write $L \in S$, to denote the disjunction $\bigvee_{s\in S} L = s$. We call $L \in S$ a *generalized linear (or quadratic) equation.*

Definition 4 (R(lin))**.** *Let $K := \{K_1, \ldots, K_m\}$ be a collection of disjunctions of linear equations in the variables $x_1, \ldots, x_n$. An* R(lin)*-proof from K of a disjunction of linear equations D is a finite sequence $\pi = (D_1, ..., D_\ell)$ of disjunctions of linear equations, such that $D_\ell = D$ and for every $i \in [\ell]$ one of the following holds:*

1. *$D_i = K_j$, for some $j \in [m]$;*
2. *D_i is a **Boolean axiom** $x_t \in \{0, 1\}$, for some $t \in [n]$;*
3. *D_i was deduced by one of the following* R(lin)*-inference rules, using D_j, D_k for some $j, k < i$:*

   Resolution *Let A, B be two, possibly empty, disjunctions of linear equations and let L_1, L_2 be two linear equations. From $A \vee L_1$ and $B \vee L_2$ derive $A \vee B \vee (L_1 - L_2)$. (We assume that every linear form with n variables is written as a sum of at most $n + 1$ monomials.[4])*

[4] Accordingly, in R(quad) we assume that every quadratic sum with n variables is written as a sum of at most $1 + 2n + \binom{n}{2}$ monomials.

Weakening *From a possibly empty disjunction of linear equations A derive $A \vee L$, where L is an arbitrary linear equation over the variables $x_1, \ldots, x_n$.*

Simplification *From $A \vee (0 = k)$ derive A, where A is a possibly empty disjunction of linear equations and $k \neq 0$.*

An R(lin) refutation of *a collection of disjunctions of linear equations K is a proof of the empty disjunction from K. The* ***size*** *of an R(lin) proof π is the total size of all the disjunctions of linear equations in π (where coefficients are written in unary representation).*

Definition 5 (R(quad))**.** *The system* R(quad) *is similar to* R(lin) *except that proof-lines can be disjunctions of quadratic equations with integer coefficients $\sum_{i,j} c_{ij} x_i x_j + \sum_i c_i x_i + c = S$ instead of linear equations; and the* ***Boolean axioms*** *are now defined for all $i, j \in [n]$, as follows:*

$$x_i \in \{0,1\}, \qquad x_i + x_j - x_i x_j \in \{0,1\}, \qquad x_i - x_i x_j \in \{0,1\}\,.$$

The size of an R(quad) *refutation is the total size of all the proof-lines in it.*

Both R(lin) and R(quad) can be proved to be sound and complete (for their respective languages, namely, disjunctions of linear and quadratic equations, respectively) refutation systems.

5.2 Comparison of the refutation system R(quad) with other systems

The R(quad) refutation system is a weak propositional proof system that, loosely speaking, can both *count* and *compose mappings*, as we explain below.

Recall that the cutting planes refutation system with small coefficients operates with linear integer inequalities of the form $\sum_i a_i x_i \geq C$ (where the a_i's are polynomial in the size of the formula to be refuted) that can be added, multiplied by a positive integer, simplified and divided by an integer c in case c divides every integer a_i, in which the division of the right hand side C/c is rounded up (i.e., we obtain $\sum_i \frac{a_i}{c} x_i \geq \lceil \frac{C}{c} \rceil$).

The cutting planes with small coefficients system can "count" to a certain extent, namely it can prove efficiently certain unsatisfiable instances encoding counting arguments (like the pigeonhole principle). However, other simple counting arguments like the Tseitin graph formulas [32] are not known to have polynomial-size cutting planes refutations.

A weak extension of cutting planes with small coefficients is defined so to allow *disjunctions* of linear equations (a big disjunction of linear equations can represent a single inequality). This way we obtain the system R(lin), that is similar to R(quad) but with *linear* instead of quadratic equations. It was shown in [28] that even when we allow disjunctions of only a *constant number* of generalized[5] linear equations in each proof-line, R(lin) has short refutations of the Tseitin formulas; this shows that using (fairly restricted) disjunctions of linear equations allows to improve the ability of cutting planes with small coefficients to refute contradictions that involve counting.

However, for our refutation of the 3XOR principle to work out we need to use quadratic instead of linear equations. Informally, the reason for this is to be able to "compose maps": as observed by Pudlák [27], the reason why the *k-Clique and $(k-1)$-Coloring* contradictions provably do not have short cutting planes refutations is that cutting planes cannot compose two mappings, which then makes it impossible to perform a routine reduction from the *k-Clique and $(k-1)$-Coloring* contradiction to the pigeonhole principle contradiction (and the latter contradiction does admit short cutting planes refutations). This is why Pudlák introduced in [27] the system CP^2 which is cutting planes operating with *quadratic inequalities*. The system R(quad) we work with is an extension of CP^2 (when the latter is restricted to small coefficients).

[5] A *generalized equation* is an equation $L \in S$, for $S \subset \mathbb{Z}$; which stands for the disjunction $\bigvee_{s \in S} L = s$.

5.3 The 3XOR principle formula

We now describe the formula Υ_n encoding the 3XOR principle (the formula depends also on the parameters t, m and k, but we will suppress these subscripts).

Recall that we wish to construct a family of formulas in three mutually disjoint sets of variables $\overline{X}, \overline{Y}, \overline{Z}$:

$$\Upsilon_n := A_n(\overline{X}, \overline{Y}) \wedge B_n(\overline{X}, \overline{Z}), \tag{5}$$

(where, in the terminology of Section 4.2, $A_n(\overline{X}, \overline{Y})$ and $B_n(\overline{X}, \overline{Z})$ are the CNF formulas expressing that $R(x, y)$ and $Q(x, z)$ are true for x of length n, respectively).

Apart from the variables $\overline{X}, \overline{Y}, \overline{Z}$ we also add a group of variables, serving as extension variables: variables that encode the product of two other variables, namely, (extension) variables that are forced to behave like products of two variables from $\overline{X}, \overline{Y}, \overline{Z}$. ***We denote such extension variables with the*** $[\![\cdot]\!]$ ***symbol;*** e.g., $[\![x_i \cdot y_j]\!]$.

Since we cannot use the $\overline{Y}$ variables in the second part of formula 5 and we cannot use the $\overline{Z}$ variables in the first part of the formula 5, we can encode only products of variables from $\overline{X}, \overline{Y}$ and from $\overline{X}, \overline{Z}$, but *not* products of a $\overline{Y}$ variable with a $\overline{Z}$ variable.

It will be convenient sometimes to denote by x_{i+n} the literal $\neg x_i$, when it is assumed we use the n variables $x_1, ..., x_n$ in the 3CNF encoded by $\overline{X}$.

A technical remark: For the sake of simplicity we *do not* encode the three unary parameters k, t, d (appearing in the disjoint **NP**-pair in Sec. 4.2) in our formula for Υ_n (and accordingly we do not encode the constraints in Equation (3)). This slightly simplifies things, and does not harm the validity of the results, as it is easy to add these constraints to the formula and give short R(quad) refutations for such a formulation.

Variables and their meaning. The variables $\overline{X}$ correspond to the input 3CNF with n variables. The variables $\overline{Y}$ correspond to the collections of t many inconsistent even k-tuples. The $\overline{Z} = \{z_1, ..., z_n\}$ variables stand for a Boolean assignment for the n variables of the 3CNF. (Note that we use the variables x_i for the variables in the 3CNF and the variables x_{ij} for the variables in our encoding of the 3CNF.)

The input 3CNF $\overline{X}$ is encoded as a $3m \times 2n$ table $\overline{X}$, where each block of three rows corresponds to a clause, and columns from 1 to n correspond to positive literals occurrences, and columns $n+1$ to $2n$ correspond to negative literals occurrences. Formally, let $1 \leq i = 3 \cdot l + r \leq 3m$, where $r \in \{0, 1, 2\}, l \in [n]$, and let $j \in [2n]$. Then $x_{ij} = 1$ means that the rth literal in the lth clause in the input 3CNF is:

$$x_j \text{ if } j \leq n, \text{ and } \neg x_{j-n}, \text{ if } j > n.$$

The collection of t inconsistent k even tuples is encoded as t tables, each table is encoded by the variables $\overline{Y}^{(s)}$, for $s \in [t]$. Each $\overline{Y}^{(s)}$ represents a table of dimension $k \times m$, where $y_{jl}^{(s)} = 1$ iff the jth member in the sth k-tuple is the lth clause (meaning the lth clause in the input 3CNF encoded by $\overline{X}$).

Group I of formulas (containing only $\overline{X}, \overline{Y}$):

1. Every row in $\overline{X}$ contains exactly one 1:

$$\sum_{j=1}^{2n} x_{ij} = 1, \quad \text{for every } i \in [3m].$$

2. Every row in $\overline{Y}^{(s)}$ contains exactly one 1:

$$\sum_{j=1}^{m} y_{ij}^{(s)} = 1, \quad \text{for all } s \in [t], i \in [k].$$

3. Every column in $\overline{Y}^{(s)}$ contains at most one 1:

$$\sum_{i=1}^{k} y_{ij}^{(s)} \in \{0,1\}, \quad \text{for all } s \in [t], j \in [m].$$

4. For any $j \in [k], r \in [m], s \in [t], \ell \in [3m], i \in [2n]$, we introduce the new **single formal variable** $[\![y_{jr}^{(s)} \cdot x_{\ell i}]\!]$ which will stand for the *product* of two other formal variables $y_{jr}^{(s)} \cdot x_{\ell i}$. For this we shall have the following axioms:

$$y_{jr}^{(s)} - [\![y_{jr}^{(s)} \cdot x_{\ell i}]\!] \in \{0,1\} \quad \text{and} \quad x_{\ell} - [\![y_{jr}^{(s)} \cdot x_{\ell i}]\!] \in \{0,1\}$$

and

$$y_{jr}^{(s)} + x_{\ell i} - [\![y_{jr}^{(s)} \cdot x_{\ell i}]\!] \in \{0,1\}$$

As an abbreviation (*not* a formal variable) we define the following:

$$Q_{ijh}^{(s)} := \sum_{r=1}^{m} [\![y_{jr}^{(s)} \cdot x_{(3(r-1)+h)i}]\!], \qquad \text{for all } i = [2n] \text{ and } h \in \{0,1,2\} \text{ and } s \in [t],$$

which expresses that x_i occurs as the hth literal in the jth clause of $\overline{Y}^{(s)}$.

5. We express that all the $\overline{Y}^{(s)}$'s are *even* k-tuples (that is, that every variable x_i appears even times) by:

$$\sum_{r \in [k], h=0,1,2} Q_{irh}^{(s)} + Q_{(i+n)rh}^{(s)} \in \{0,2,4,...,k\}, \qquad \text{for all } i \in [n], s \in [t].$$

We can assume that k is even, since for every even k-tuple k must be even.

6. Similarly, we encode that the $\overline{Y}^{(s)}$'s are *inconsistent* (that is, the number of negative literals in them is odd) by:

$$\sum_{\substack{r \in [k], h=0,1,2 \\ i \in [n]}} Q_{(i+n)rh}^{(s)} \in \{0,3,5,...,k-1\}.$$

7. Every clause $i \in [m]$ appears in at most d even k-tuples $\overline{Y}^{(1)}, ..., \overline{Y}^{(t)}$. We put:

$$\sum_{j \in [k], s \in [t]} y_{ji}^{(s)} \in \{0,1,...,d\}, \qquad \text{for every } i \in [m].$$

This finishes the encoding of the t inconsistent even k-tuples.

Group II of formulas (containing only $\overline{X}, \overline{Z}$): We now turn to the formulas expressing that there are assignments $\overline{Z}$ that satisfy more than $m - \lceil t/d \rceil$ clauses in $\overline{X}$ as 3XORs. For every $j \in [3m], i \in [2n], \ell \in [n]$, let $[\![x_{ji} \cdot z_\ell]\!]$ be a new formal variable that stands for the product $x_{ji} \cdot z_\ell$. As in part 4 of the formula above, we include the axioms that force $[\![x_{ji} \cdot z_\ell]\!]$ to stand for $x_{ji} \cdot z_\ell$.

Let us use the following abbreviation:

$$U_j := \sum_{h=0,1,2} \left(\sum_{i=1}^{n} [\![x_{(3(j-1)+h)i} \cdot z_i]\!] + \sum_{i=1}^{n} \left(x_{(3(j-1)+h)(i+n)} - [\![x_{(3(j-1)+h)(i+n)} \cdot z_i]\!] \right) \right).$$

Then, $U_j \in \{1,3\}$ states that the jth clause in $\overline{X}$ is satisfied as 3XOR by $\overline{Z}$. Note that $x_{(3(j-1)+h)(i+n)} - [\![x_{(3(j-1)+h)(i+n)} \cdot z_i]\!]$ is a linear term that expresses the quadratic term $x_{(3(j-1)+h)(i+n)} \cdot (1 - z_i)$.

8. Let u_j be a new formal variable expressing that the jth clause in $\overline{X}$ is satisfied as 3XOR by $\overline{Z}$. Hence, $U_j \in \{1,3\}$ iff $u_j = 1$, and we encode it as:

$$U_j \in \{0,2\} \vee (u_j = 1) \quad \text{and} \quad U_j \in \{1,3\} \vee (u_j = 0),$$

9. There are assignments $\overline{Z}$ that satisfy more than $m - \lceil t/d \rceil$ clauses in $\overline{X}$ as 3XORs:

$$\sum_{j=1}^{m} u_j \in \{m - \lceil t/d \rceil + 1, ..., m\}.$$

The set of formulas described in this section has no $0,1$ solution by virtue of the 3XOR principle itself (Section 4.1).

6 Short refutations for the 3XOR principle

In this section we demonstrate polynomial-size (in n) R(quad) refutations of the 3XOR principle as encoded by disjunctions of linear equations in the previous section.

Theorem 2. R(quad) *admits polynomial-size refutations of the 3XOR principle formulas.*

We sometimes give only a high level description of the derivations. We use the terminology and abbreviations in Section 5. We also use freely the ability of R(lin) (and hence R(quad)) to count. For a detailed treatment of efficient counting arguments inside R(lin) see [28].

Step 1: Working in R(quad), we first show that our axioms prove that $\overline{Z}$ cannot satisfy as 3XOR all clauses of $\overline{Y}^{(s)}$, for any $s \in [t]$.

Recall from Section 5 the abbreviation

$$Q^{(s)}_{ijh} := \sum_{r=1}^{m} [\![y^{(s)}_{jr} \cdot x_{(3(r-1)+h)i}]\!], \qquad \text{for all } i = [2n] \text{ and } h \in \{0,1,2\} \text{ and } s \in [t],$$

which stands for the statement that x_i occurs as the hth literal in the jth clause of $\overline{Y}^{(s)}$ (and where x_i for $i > n$ stands for the literal $\neg x_{i-n}$). Let us use the abbreviation:

$$P_{jhs} := \sum_{i=1}^{n} Q^{(s)}_{ijh} \cdot z_i + \sum_{i=1}^{n} Q^{(s)}_{(i+n)jh} \cdot (1 - z_i).$$

Then, P_{jhs} is a quadratic sum that stands for the statement that the hth literal in clause j in $\overline{Y}^{(s)}$ is true under $\overline{Z}$. Thus,

$$P_{j0s} + P_{j1s} + P_{j2s} \in \{1, 3\}, \quad \text{for all } j \in [k] \tag{6}$$

expresses that all the clauses in $\overline{Y}^{(s)}$ are satisfied as 3XOR under $\overline{Z}$.

Our goal now is to refute (6), based on our axioms. Informally, this refutation is done by counting: first count by clauses in $\overline{Y}^{(s)}$, namely, add all left hand sides of (6) together reaching an even number (in the right hand side) by virtue of k being even (recall we can assume that k is even). Then, count by literals, namely sum all values of literals in $Y^{(s)}$ under the assignment $\overline{Z}$, which we can prove is odd from our axioms. We now describe this refutation more formally.

Since k is even, counting by clauses in $\overline{Y}^{(s)}$, namely, adding the left hand sides of (6) gives us easily the following (with a polynomial-size R(quad) proof):

$$\sum_{j=1}^{k} P_{j0s} + P_{j1s} + P_{j2s} \in \{0, 2, 4, ..., 3k\}. \tag{7}$$

Now we need to count by literals in $\overline{Y}^{(s)}$. We can abbreviate the number of occurrences in $\overline{Y}^{(s)}$ of the literal x_i, for $i \in [2n], s \in [t]$, by:

$$T_i := \sum_{\substack{j \in [k] \\ h=0,1,2}} Q^{(s)}_{ijh}.$$

Let us abbreviate by S_i the contribution of the literals x_i and $\neg x_i$ to the total sum (7). Thus

$$S_i := \sum_{\substack{j \in [k] \\ h=0,1,2}} Q^{(s)}_{ijh} \cdot z_i + \sum_{\substack{j \in [k] \\ h=0,1,2}} Q^{(s)}_{(i+n)jh} \cdot (1 - z_i).$$

It is possible to prove the following:

$$T_i \in \{0, 2, 4, ..., k\} \vee S_i \in \{1, 3, 5, ..., k-1\} \tag{8}$$

which states that if the number of occurrences in $\overline{Y}^{(s)}$ of the literal x_i is odd then (since by our axioms stating that every variable occurs even times, the number of occurrences of the literal $\neg x_i$ must also be odd) the contribution of x_i and $\neg x_i$ to the total sum (7) is also odd (because either $z_i = 0$ or $z_i = 1$).

By the axioms saying that the number of negative literals is odd (axiom 6) we get that:

$$\sum_{i=1}^{n} T_{i+n} \in \{1, 3, 5, ..., k \cdot n - 1\}. \tag{9}$$

And from the axioms stating that each variable occurs even times in $\overline{Y}^{(s)}$ we have:

$$T_i + T_{i+n} \in \{0, 2, 4, ..., k\}, \quad \text{for all } i \in [n]. \tag{10}$$

From (10) we obtain $\sum_{i=1}^{2n} T_i \in \{0, 2, 4, ..., k \cdot n\}$, and from this and (9) we obtain

$$\sum_{i=1}^{n} T_i \in \{1, 3, 5, ..., k \cdot n - 1\}. \tag{11}$$

Note that (8) can be interpreted as saying that if T_i is odd then so does S_i. Accordingly, one can use (8) to substitute all $T_1, ..., T_n$ in (11) with $S_1, ..., S_n$, respectively. We thus get that the total sum in the left hand side of (7) is in $\{1, 3, 5, ...\}$, and we obtain a contradiction with (7).

From a refutation of the collection of disjunctions (6), for any $s \in [t]$, we can actually get the negation of this collection, that is:

$$\bigvee_{j\in[k]} (P_{j0s} + P_{j1s} + P_{j2s}) \in \{0, 2\}. \tag{12}$$

This stems from the following: it is already true in resolution that if we have a size γ resolution refutation of $A_1, ..., A_l$, then assuming the axioms $A_1 \vee B_1, ..., A_l \vee B_l$, we can have a size $O(\gamma \cdot d)$ resolution derivation of $B_1 \vee ... \vee B_l$, given that the total size of the B_i's is d. To see this, take the resolution refutation of $A_1, ..., A_l$ and OR every line in this refutation with $B_1 \vee ... \vee B_l$ (note that the resulting new axioms are actually derivable from the axioms $A_i \vee B_i$ via Weakening). Now, to get (12) from (6), we do the same, putting $P_{j0s} + P_{j1s} + P_{j2s} \in \{1, 3\}$ instead of A_j and $P_{j0s} + P_{j1s} + P_{j2s} \in \{0, 2\}$ instead of B_j, for all $j \in [k]$, noting that:

$$(P_{j0s} + P_{j1s} + P_{j2s} \in \{1, 3\}) \vee (P_{j0s} + P_{j1s} + P_{j2s} \in \{0, 2\}), \quad \text{for all } j \in [k]. \tag{13}$$

Step 2: The next step in our R(quad) refutation is showing how to obtain the final contradiction, given the collection of formulas (12), for all $s \in [t]$. This is again by counting: we know that for every truth assignment $\overline{Z}$, each $Y^{(1)}, ..., Y^{(t)}$ must contribute at least one clause from $\overline{X}$ that is unsatisfiable as 3XOR under $\overline{Z}$. We can view this as a mapping $g : [t] \to [m]$ from $Y^{(1)}, ..., Y^{(t)}$ to the m clauses in $\overline{X}$, such that $g(i) = j$ means that $Y^{(i)}$ contributes the clause j in $\overline{X}$ that is unsatisfiable under $\overline{Z}$ as 3XOR. The mapping g is not 1-to-1, but d-to-1, because every clause of $\overline{X}$ can appear at most d times in $Y^{(1)}, ..., Y^{(s)}$. Our R(quad) refutation proceeds as follows.

By assumption we have $\sum_{i=1}^{m} u_i \in \{m - \lceil t/d \rceil + 1, ..., m\}$, meaning that the number of clauses in $\overline{X}$ that are satisfied as 3XOR under the assignment $\overline{Z}$ is at least $m - \lceil t/d \rceil + 1$. Also, by the axioms in our formula, for all $i \in [m]$ we can prove that $u_i = 1$ implies that $U_i \in \{1, 3\}$; namely that the number of true literals in the ith clause of $\overline{X}$ is 1 or 3.

For any $s \in [t]$, we can think of $\overline{Y}^{(s)}$ as a mapping $f^{(s)} : [k] \to [m]$ that maps the k clauses in $\overline{Y}^{(s)}$ to the clauses in $\overline{X}$. Then, $y_{ij}^{(s)} = 1$ means that $f^{(s)}(i) = j$. Thus, $u_i \cdot y_{ji}^{(s)} = 1$ means that the jth clause in $\overline{Y}^{(s)}$ is the ith clause in $\overline{X}$ and that the ith clause in $\overline{X}$ is satisfiable as 3XOR under $\overline{Z}$.

Now, it is possible to show that for any $s \in [t]$, $i \in [m]$ and $j \in [k]$, there is a proof of the following line:

$$\left(u_i \cdot y_{ji}^{(s)} = 0\right) \vee (P_{j0s} + P_{j1s} + P_{j2s} \in \{1, 3\}) \tag{14}$$

which states that if the ith clause in $\overline{X}$ is satisfied as 3XOR under the assignment $\overline{Z}$ and the jth clause in $\overline{Y}^{(s)}$ maps to the ith clause in $\overline{X}$, then the jth clause in $\overline{Y}^{(s)}$ is satisfied as 3XOR under $\overline{Z}$.

Informally, the proof of (14) is explained as follows: the term $P_{j0s} + P_{j1s} + P_{j2s}$ can be seen as the addition, denoted $\mathcal{S}$, of all inner products of the jth row of $\overline{Y}^{(s)}$ with the columns of $\overline{X}$ (for each $h = 0, 1, 2$ we can consider the column of $\overline{X}$ restricted to the $h \cdot i$ rows only ($i \in [m]$), and so a row of $\overline{Y}^{(s)}$ which is of length m can have an inner product with such a column of length m in $\overline{X}$). Because we assume that $y_{ij}^{(s)} = 1$, only the ith coordinate in the jth row of $\overline{Y}^{(s)}$ is 1 (and all the other entries in this row are 0, by our axioms). Thus, $\mathcal{S}$ equals in fact a single column from $\overline{X}$; and this single column is precisely U_i.

From (14) and (12) we can derive, for any $s \in [t]$ and any $i \in [m]$:

$$\bigvee_{j\in[k], i\in[m]} \left(y_{ji}^{(s)} \cdot (1 - u_i) = 1\right), \tag{15}$$

stating that for some $j \in [k], i \in [m]$, the jth clause in $\overline{Y}^{(s)}$ is the ith clause in $\overline{X}$ and the ith clause in $\overline{X}$ is not satisfied as 3XOR under $\overline{Z}$.

Now, from (15) and axioms 7 in the 3XOR principle formulas, stating that $g : [t] \to [m]$ is d-to-1, we can obtain that the number of u_i's that are true is no more than $m - \lceil t/d \rceil$, that is, $\sum_{i\in[m]} u_i \in \{0, ..., m - \lceil t/d \rceil\}$, contradicting the axiom $\sum_{j=1}^{m} u_j \in \{m - \lceil t/d \rceil + 1, ..., m\}$. The formal proofs of this in R(quad) is shown in the following lemma:

Lemma 9. *There are polynomial-size* R(quad) *refutations of (15) and the axioms in parts 7 and 9 in the 3XOR principle.*

Proof. First sum all axioms (7) to obtain:

$$\sum_{\substack{j\in[k],s\in[t]\\ r\in[m]}} y_{jr}^{(s)} \in \{0, 1, ..., d \cdot m\}. \tag{16}$$

From (15) we can obtain:

$$\sum_{j\in[k],r\in[m]} y_{jr}^{(s)} \cdot (1 - u_i) \in \{1, 2, ..., k \cdot m\}, \qquad \text{for every } s \in [t].$$

And by summing this for all $s \in [t]$ and $i \in [m]$, we get:

$$\sum_{i\in[m]} \sum_{\substack{j\in[k],r\in[m]\\ s\in[t]}} y_{jr}^{(s)} \cdot (1 - u_i) = \sum_{i\in[m]} (1 - u_i) \cdot \sum_{\substack{j\in[k],r\in[m]\\ s\in[t]}} y_{jr}^{(s)}$$
$$\in \{t \cdot m, t \cdot m + 1, ..., t \cdot k \cdot m^2\}. \tag{17}$$

From the axiom in part (9) in the 3XOR principle $\sum_{j=1}^{m} u_j \in \{m - \lceil t/d \rceil + 1, ..., m\}$ we can obtain easily

$$\sum_{i\in[m]} (1 - u_i) \in \{0, 1, ..., \lceil t/d \rceil - 1\}.$$

From this and (16) we get, via Lemma 10 proved below, the following:

$$\sum_{i\in[m]} (1 - u_i) \cdot \sum_{\substack{j\in[k],s\in[t]\\ r\in[m]}} y_{jr}^{(s)} \in \{0, 1, ..., d \cdot m \cdot (\lceil t/d \rceil - 1)\}.$$

Since $d \cdot m \cdot (\lceil t/d \rceil - 1) < d \cdot m \cdot \lceil t/d \rceil \leq m \cdot t$, we obtain a contradiction with (17), which finishes the refutation. QED

It remains to prove Lemma 10, which was used in the above proof:

Lemma 10. *Let $\sum_{i\in I} x_i \in \{0, 1, ..., n\}$ and $\sum_{j\in J} y_j \in \{0, 1, ..., m\}$ be disjunctions of linear equations, both of size at most s. Given these two disjunctions we can prove in* R(quad) *with a polynomial-size in s proof, the following:*

$$\sum_{i\in I} x_i \cdot \sum_{j\in J} y_j \in \{0, 1, ..., m \cdot n\}. \tag{18}$$

Proof. We can reason in a case-by-case manner as follows (see [28] on how to carry out informal case-analysis reasoning inside R(lin)): assume that $\sum_{j\in J} y_j = a$, for $a \in \{0, 1, ..., m\}$. We wish to show that $x_1 \cdot \sum_{j\in J} y_j = ax_1$. If $x_1 = 0$ then $x_1 \cdot \sum_{j\in J} y_j = 0 = ax_1$. Otherwise, $x_1 = 1$. Then, $x_1 \cdot \sum_{j\in J} y_j = \sum_{j\in J} y_j = a = ax_1$. Since we have the axiom $(x_1 = 0) \vee (x_1 = 1)$ we conclude that $x_1 \cdot \sum_{j\in J} y_j = ax_1$. In a similar way we can derive for all $i \in I$:

$$x_i \cdot \sum_{j\in J} y_j = ax_i. \tag{19}$$

And by adding (19) for all $i \in I$ we obtain:

$$\sum_{i\in I} x_i \cdot \sum_{j\in J} y_j = a \cdot \sum_{i\in I} x_i.$$

Now using the axiom $\sum_{i\in I} x_i \in \{0, 1, ..., n\}$, we get

$$\sum_{i\in I} x_i \cdot \sum_{j\in J} y_j \in \{0, a, 2a, ..., n \cdot a\}. \tag{20}$$

Recall that (20) was obtained under the assumption that $\sum_{j\in J} y_j = a$. This means that if we have the axiom $\sum_{j\in J} y_j \in \{0, 1, ..., m\}$, we can obtain:

$$\sum_{i\in I} x_i \cdot \sum_{j\in J} y_j \in \{b \cdot c \mid b \in \{0, 1, .., n\} \text{ and } c \in \{0, 1, ..., m\}\} = \{0, 1, ..., n \cdot m\}.$$

QED

Note that the proof of Lemma 10 would also work if instead of the sums $\sum_{i\in I} x_i$ or $\sum_{j\in J} y_j$ we have $\sum_{i\in I} b_i x_i$ or $\sum_{j\in J} c_j y_j$, for integers b_i, c_j.

7 Reduction to weak automatizability of R(lin)

Here we show that R(lin) is weakly automatizable if and only if R(quad) is weakly automatizable.

To show that R(lin) is weakly automatizable iff R(quad) is weakly automatizable we use a similar idea to Pudlák [27]. Namely, we show that the *canonical pair* of R(quad) is polynomially reducible to the canonical pair of R(lin).

Definition 6 ([29])**.** *The* canonical pair *of a refutation system $\mathcal{P}$ is the disjoint **NP**-pair, whose first **NP** language consists of all pairs $(\tau, 1^m)$ where τ is an unsatisfiable formula that has a $\mathcal{P}$-refutation of size at most m, and whose second **NP** language is the set of pairs $(\mu, 1^m)$ where μ is a satisfiable formula and m is some natural number.*

We say that a canonical pair (A, B) of a refutation system $\mathcal{P}'$ is *polynomially reducible* to the canonical pair (A', B') of another refutation system $\mathcal{P}$ if there is a polynomial-time computable function f such that for all x it holds that $x \in A \iff f(x) \in A'$ and $x \in B \iff f(x) \in B'$. A simple corollary of the above definitions is the following:

Proposition 11 ([27])**.** *If the canonical pair of $\mathcal{P}'$ is polynomially reducible to the canonical pair of $\mathcal{P}$ then $\mathcal{P}'$ is weakly automatizable if $\mathcal{P}$ is weakly automatizable.*

In view of this proposition, and since R(quad) clearly polynomially simulates R(lin) (as an extension of it), it remains to show the following:

Proposition 12. *The canonical pair of* R(quad) *is polynomially reducible to the canonical pair of* R(lin)*.*

Proof. (Sketch) Similar to [27], the idea is to encode a product of any two variables $x_i \cdot x_j$ as a new single formal variable x_{ij}. Thus, the reduction sends all pairs $(\tau, 1^m)$ to the pair $(\tau', 1^{\text{poly}(m)})$, where τ' is obtained from τ by adding the axioms that force all new variables x_{ij} to encode the product $x_i \cdot x_j$, as shown in Section 5. QED

Corollary 4. R(quad) *is weakly automatizable iff* R(lin) *is weakly automatizable.*

Since R(quad) admits polynomial-size refutations of the 3XOR principle, and since weak automatizability entails feasible interpolation, we get a reduction of the problem of determinizing Feige et al. nondeterministic refutation algorithm to the problem of establishing weak automatizability of R(lin):

Corollary 5. *If* R(lin) *is weakly automatizable then there is a deterministic refutation algorithm for random 3CNFs with $\Omega(n^{1.4})$ clauses.*

Acknowledgments

I wish to thank Jan Krajíček for useful comments related to this work and Albert Atserias and Neil Thapen for useful related discussions.

References

[1] Dimitris Achlioptas. Random satisfiability. In *Handbook of Satisfiability*, pages 245–270. 2009. 1

[2] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM J. Comput.*, 38(4):1347–1363, 2008. 3

[3] A. Atserias and Maria Luisa Bonet. On the automatizability of resolution and related propositional proof systems. *Information and Computation*, 189:182–201, 2004. 1.1, 1.1, 1.2

[4] Albert Atserias and Elitza Maneva. Mean-payoff games and propositional proofs. In *International Conference on Automata, Languages and Programming*, volume 6198 of *Lecture Notes in Computer Science*, pages 102–113. Springer Berlin / Heidelberg, 2012. 1.2

[5] Paul Beame and Toniann Pitassi. Propositional proof complexity: past, present, and future. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (65):66–89, 1998. 2.1

[6] Arnold Beckmann, Pavel Pudlák, and Neil Thapen. Parity games and propositional proofs. *ACM Transactions on Computational Logic*. To appear. 1.2

[7] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997. 1.1, 1.1, 2

[8] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for Frege systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000. 1.1, 1.1, 1.2

[9] Peter Clote and Evangelos Kranakis. *Boolean functions and computation models*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2002. 2.1

[10] Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong refutation heuristics for random k-SAT. *Combinatorics, Probability & Computing*, 16(1):5–28, 2007. 1

[11] W. Cook, C. R. Coullard, and G. Turan. On the complexity of cutting plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987. 1.1

[12] Uriel Feige. Relations between average case complexity and approximation complexity. In *STOC*, pages 534–543, 2002. 4.1

[13] Uriel Feige. Refuting smoothed 3CNF formulas. In *Proceedings of the IEEE 48th Annual Symposium on Foundations of Computer Science*, pages 407–417. IEEE Computer Society, 2007. 1

[14] Uriel Feige, Jeong Han Kim, and Eran Ofek. Witnesses for non-satisfiability of dense random 3CNF formulas. In *Proceedings of the IEEE 47th Annual Symposium on Foundations of Computer Science*, 2006. (document), 1, 1.2, 3, 3, 6, 4.1, 4.1, 7, 4.1

[15] Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3CNF formulas. *Theory of Computing*, 3(1):25–43, 2007. (document), 1, 4.1

[16] Joel Friedman, Andreas Goerdt, and Michael Krivelevich. Recognizing more unsatisfiable random k-SAT instances efficiently. *SIAM J. Comput.*, 35(2):408–430, 2005. 1

[17] A. Goerdt and M. Krivelevich. Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 294–304, 2001. 1

[18] Andreas Goerdt and André Lanka. Recognizing more random unsatisfiable 3-SAT instances efficiently. *Electronic Notes in Discrete Mathematics*, 16:21–46, 2003. 1

[19] Lei Huang and Toniann Pitassi. Automatizability and simple stochastic games. In *ICALP (1)*, pages 605–617, 2011. 1.2

[20] Jan Krajíček. Lower bounds to the size of constant-depth propositional proofs. *The Journal of Symbolic Logic*, 59(1):73–86, 1994. 1.1

[21] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1995. 2.1

[22] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997. 1.1

[23] Jan Krajíček. On the weak pigeonhole principle. *Fund. Math.*, 170(1-2):123–140, 2001. 1.1

[24] Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Inform. and Comput.*, 140(1):82–94, 1998. 1.1

[25] Sebastian Müller and Iddo Tzameret. Short propositional refutations for dense random 3CNF formulas. In *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic In Computer Science (LICS)*, 2012. *Annals of Pure and Applied Logic* (accepted subject to minor revisions), 2013. 1.1, 1.2, 4.1

[26] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, Sept. 1997. 1.1

[27] Pavel Pudlák. On reducibility and symmetry of disjoint NP pairs. *Theoret. Comput. Sci.*, 295:323–339, 2003. 1.1, 1.1, 1.1, A.2, C, 11, C

[28] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. (document), 1.1, A.1, A.2, B, B

[29] Alexander A. Razborov. On provably disjoint NP-pairs. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(6), 1994. 1.1, 6

[30] Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izv. Ross. Akad. Nauk Ser. Mat.*, 59(1):201–224, 1995. 1.1

[31] Nathan Segerlind. The complexity of propositional proofs. *Bull. Symbolic Logic*, 13(4):417–481, 2007. 2.1

[32] Grigori Tseitin. *On the complexity of derivations in propositional calculus.* Studies in constructive mathematics and mathematical logic Part II. Consultants Bureau, New-York-London, 1968. A.2