Lower bounds for splittings by linear combinations *

Dmitry Itsykson† and Dmitry Sokolov†

August 6, 2014

Abstract

A typical DPLL algorithm for the Boolean satisfiability problem splits the input problem into two by assigning the two possible values to a variable; then it simplifies the two resulting formulas. In this paper we consider an extension of the DPLL paradigm. Our algorithms can split by an arbitrary linear combination of variables modulo two. These algorithms quickly solve formulas that explicitly encode linear systems modulo two, which were used for proving exponential lower bounds for conventional DPLL algorithms.

We prove exponential lower bounds on the running time of DPLL with splitting by linear combinations on 2-fold Tseitin formulas and on formulas that encode the pigeonhole principle.

Raz and Tzameret introduced a system $R(lin)$ which operates with disjunctions of linear equalities with *integer coefficients*. We consider an extension of the resolution proof system that operates with disjunctions of linear equalities over $\mathbb{F}_2$; we call this system Res-Lin. Res-Lin can be p-simulated in $R(lin)$ but currently we do not know any superpolynomial lower bounds in $R(lin)$. Tree-like proofs in Res-Lin are equivalent to the behavior of our algorithms on unsatisfiable instances. We prove that Res-Lin is implication complete and also prove that Res-Lin is polynomially equivalent to its semantic version. We prove a space-size tradeoff for Res-Lin proofs of 2-fold Tseitin formulas.

1 Introduction

Splitting is the one of the most frequent methods for exact algorithms for NP-hard problems. It considers several cases and recursively executes on each of that cases. For the CNF satisfiability problem the classical splitting algorithms are so called DPLL algorithms (by authors Davis, Putnam, Logemann and Loveland) [6], [5] in which splitting cases are values of a variable. A very natural extension of such algorithms is a splitting by a value of some formula. In this paper we consider an extension of DPLL that allows splitting by linear combinations of variables over $\mathbb{F}_2$. There is a polynomial time

*The research is partially supported by the RFBR grant 14-01-00545, by the President's grant MK-2813.2014.1 and by the Government of the Russia (grant 14.Z50.31.0030).

†Steklov Institute of Mathematics at St. Petersburg, 27 Fontanka, St.Petersburg, 191023, Russia, dmitrits@pdmi.ras.ru, sokolov.dmt@gmail.com.

algorithm that check whether a system of linear equations has a solution and whether a system of linear equations contradicts a clause. Thus the running time of an algorithm that solves CNF-SAT using splitting by linear combinations (in the contrast to splitting by arbitrary functions) is at most the size of its splitting tree up to a polynomial factor.

Formulas that encode unsatisfiable systems of linear equations are hard for resolution and hence for DPLL [16], [3]. Systems of linear equations are also hard satisfiable examples for myopic and drunken DPLL algorithms [1], [9]. Hard examples for myopic algorithms with a cut heuristic are also based on linear systems [8]. We show that a splitting by linear combinations helps to solve explicitly encoded linear systems over $\mathbb{F}_2$ in polynomial time.

For every CNF formula ϕ we denote by $\phi^{\oplus}$ a CNF formula obtained from ϕ by substituting $x_1 \oplus x_2$ for each variable x. Urquhart shows that for unsatisfiable ϕ the running time of any DPLL algorithm on $\phi^{\oplus}$ is at least $2^{d(\phi)}$, where $d(\phi)$ is the minimal depth of the recursion tree of DPLL algorithms running on the input ϕ [17]. Urquhart also gives an example of Pebbling contradictions $Peb(G_n)$ such that $d(Peb(G_n)) = \Omega(n/\log n)$ and there is a DPLL algorithm that solves $Peb(G_n)$ in $O(n)$ steps. Thus $Peb^{\oplus}(G_n)$ is one more example that is hard for DPLL algorithms but easy for DPLL with splitting by linear combinations.

The recent algorithm by Seto and Tamaki [15] solves satisfiability of formulas over full binary basis using a splitting by linear combination of variables. The similar idea was used by Demenkov and Kulikov in the simplified lower bound $3n - o(n)$ for circuit complexity over full binary basis [7]. The common idea of [15] and [7] is that a restricting a circuit with a linear equation may significantly reduce the size of the circuit.

Our results. We prove an exponential lower bound on the size of a splitting tree by linear combinations for 2-fold Tseitin formulas that can be obtained from ordinary Tseitin formulas by substituting every variable by the conjunction of two new variables. The plan of the proof is following: let for every unsatisfiable formula ϕ a search problem $Search_\phi$ be the problem of finding falsified clause given a variable assignment. We prove that it is possible to transform a splitting tree T into a randomized communication protocol for the problem $Search_\phi$ of depth $O(\log |T| \log\log |T|)$ if some variables are known by Alice and other variables are known by Bob. And finally, we note that a lower bound on the randomized communication complexity of the problem $Search_{TS^2_{G,c}}$ for a 2-fold Tseitin formula $TS^2_{G,c}$ follows from [10] and [2].

We also give an elementary proof of the lower bound $2^{\frac{n-1}{2}}$ on the size of linear splitting trees of formulas PHP^m_n that encode the pigeonhole principle. We also show that the formulas $PM(K_{n,n+1})$ that code the existence of perfect mathcing in the complete bipartite graph $K_{n,n+1}$ has polynomial size linear splitting trees while $PM(K_{n,n+1})$ are exponentially hard for resolution[14].

It is well known that the behavior of DPLL algorithms on unsatisfiable formulas corresponds to tree-like resolution proofs. We consider the extension of the resolution proof system that operates with disjunctions of linear equalities. A system Res-Lin contains the weakening rule and the resolution rule. We also consider a system Sem-Lin that is a semantic version of Res-Lin; Sem-Lin contains semantic implication rule with two premises instead of the resolution rule. We prove that this two systems are

polynomially equivalent and they are implication complete. We also show that tree-like versions of Res-Lin and Sem-Lin are equivalent to linear splitting trees; the latter implies that our lower bounds hold for tree-like Res-Lin and Sem-Lin. Raz and Tzameret studied a system $R(lin)$ which operates with disjunctions of linear equalities with *integer coefficients* [12]. It is possible to p-simulate Res-Lin in $R(lin)$ but the existence of the simulation in the other direction is an open problem. We also prove a space-size tradeoff for Res-Lin proofs of 2-fold Tseitin formulas.

Futher research. The main open problem is to prove a superpolynomial lower bound in the DAG-like Res-Lin. One of the ways to prove a lower bound is to simulate the Res-Lin system with another system for which a superpolynomial lower bound is known. It is impossible to simulate Res-Lin with Res(k) (that extends Resolution and operate with k-DNF instead of clauses) and in PCR (Polynomial Calculus + Resolution) over field with $char \neq 2$ because there are known exponential lower bounds in Res(k) and PCR for formulas based on systems of linear equations [13]. It is interesting whether it is possible to simulate Res-Lin with Polynomial Calculus (or PCR) over $\mathbb{F}_2$ or with the system $R^0(lin)$ which is a subsystem of $R(lin)$ with known exponential lower bounds based on the interpolation. Another open problem is to prove lower bounds for splitting by linear combinations on satisfiable formulas, for example, for algorithms that arbitrary choose a linear combination for splitting and randomly choose a value to investigate first.

2 Preliminaries

We will use the following notation: $[n] = \{1, 2, \ldots, n\}$. Let $X = \{x_1, \ldots, x_n\}$ be a set of variables that take values from $\mathbb{F}_2$. A linear form is a polynomial $\sum_{i=1}^{n} \alpha_i x_i$ over $\mathbb{F}_2$.

Consider a binary tree T with edges labeled with linear equalities. For every vertex v of T we denote by Φ_v^T a system of all equalities that are written along the path from the root of T to v. *A linear splitting tree* for a CNF formula ϕ is a binary tree T with the following properties. Every internal node is labeled by a linear form that depends on variables from ϕ. For every internal node that is labeled by a linear form f one of the edges going to its children is labeled by $f = 0$ and the other edge is labeled by $f = 1$. For every leaf v of the tree exactly one of the following conditions hold: 1) The system Φ_v^T does not have solutions. We call such leaf degenerate. 2) The system Φ_v^T is satisfiable but contradicts a clause C of formula ϕ. We say that such leaf refutes C. 3) The system Φ_v^T has exactly one solution in the variables of ϕ and this solution satisfies the formula ϕ. We call such leaf satisfying.

A linear splitting tree may also be viewed as a recursion tree of an algorithm that searches for satisfying assignments of a CNF formula using the following recursive procedure. It gets on the input a CNF formula ϕ and a system of linear equations Φ, the goal of the algorithm is to find a satisfying assignment of $\phi \wedge \Phi$. Initially $\Phi = True$ and on every step it somehow chooses a linear form f and a value $\alpha \in \mathbb{F}_2$ and makes two recursive calls: on the input $(\phi, \Phi \wedge (f = \alpha))$ and on the input $(\phi, \Phi \wedge (f = 1 + \alpha))$. The algorithm backtracks in one of the three cases: 1) The system Φ does not have solutions (it can be verified in polynomial time); 2) The system Φ contradicts to a clause C of the formula ϕ (A system Ψ contradicts a clause $(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k)$ iff for all $i \in [k]$ the

system $\Psi \wedge (\ell_i = 1)$ is unsatisfiable. Hence this condition may be verified in polynomial time); 3) The system Φ has the unique solution that satisfies ϕ (it can also be verified in polynomial time). Note that if it is enough to find just one satisfying assignment, then the algorithm may stop in the first satisfying leaf. But in the case of unsatisfiable formulas it must traverse the whole splitting tree.

Proposition 2.1. For every linear splitting tree T for a formula ϕ it is possible to construct a splitting tree that has no degenerate leaves. The number of vertices in the new tree is at most the number of vertices in T.

Proof. Let T contain vertices v (not necessary leaves) such that Φ_v^T is unsatisfiable. Let w be the one closest to the root. The vertex w differs from the root because the system in the root is empty and therefore satisfiable. Let s be the parent of w and u is the brother of v. The system Φ_u^T is not unsatisfiable since in the opposite case Φ_s^T is also unsatisfiable that contradicts the choice of w. We construct the new tree T' from T by removing a subtree with the root w and contracting the edge (s, u). T' is a correct splitting tree because for all nodes v in T' a system $\Phi_v^{T'}$ is equivalent to the system in the corresponding vertex of T. We continue applying this transformation while the tree changes. □

3 Upper bounds

Proposition 3.1. Let formula ϕ in CNF encode an unsatisfiable system of linear equations $\bigwedge_{i=1}^m (f_i = \beta_i)$ over $\mathbb{F}_2$. The i-th equation $f_i = \beta_i$ is represented by a CNF formula ϕ_i and $\phi = \bigwedge_{i=1}^m \phi_i$. It is possible that encodings of different formulas ϕ_i have the same clause; we assume that such clause is repeated in ϕ. Then there exists a splitting tree for ϕ of size $O(|\phi|)$.

Proof. We will describe a binary tree T that has a path from the root to a leaf labeled by equalities $f_1 = \beta_1, f_2 = \beta_2, \ldots, f_m = \beta_m$; the leaf is degenerate since the system is unsatisfiable. For all $i \in [m]$ the i-th vertex on the path has the edge to the child u_i labeled by $f_i = \beta_i + 1$. Now we describe a subtree T_{u_i} with the root u_i; it is just a splitting tree over all variables of formula ϕ_i. Let $x_1, x_2, \ldots, x_k$ be variables that appear in f with nonzero coefficients. We sequentially make splittings on $x_1, x_2, \ldots x_k$ starting in u_i. We know that $\Phi_{u_i}^T$ contradicts ϕ_i, therefore every leaf of T_{u_i} either refutes clause of ϕ_i or is degenerate (the system contradicts $f_i = 1 + \beta_i$). T_{u_i} has 2^k leaves but it is well known that every CNF representation of $x_1 + x_2 + \cdots + x_k = \beta_i$ has at least 2^{k-1} clauses. Therefore the size of T is at most $O(|\phi|)$. □

For every graph $G(V, E)$ we define a formula $PM(G)$ that encode the existence of the perfect matching in G. Every edge $e \in E$ corresponds to a variable x_e. For every vertex $v \in V$ the formula $PM(G)$ contains a clause $\bigvee_{(v,u)\in E} x_{(v,u)}$ and for every pair of edges (v, u) and $(v, w) \in E$ it contains a clause $\neg x_{(v,u)} \vee \neg x_{(v,w)}$. The formula $PM(G)$ is satisfiable of and only if G has a perfect matching.

Proposition 3.2. If $G(V, E)$ has odd number of vertices, then there exists a polynomial size linear splitting tree for $PM(G)$.

Proof. We will describe a binary tree T that has a path from the root to a leaf labeled by equalities $\sum_{(v,u)\in E} x_{(v,u)} = 1$ for all $v \in V$; the leaf is degenerate since the number of vertices is odd and hence the sum of all equalities along the path is a contradiction $0 = 1$. For all i the i-th vertex on the path has the edge to the child u_i labeled by $\sum_{(v_i,u)\in E} x_{(v_i,u)} = 0$. Now we describe a subtree T_{u_i} with the root u_i; it is a splitting tree over all variables $x_{v_i,u}$ such that $(v_i, u) \in E$. Whenever we substitute 1 for two variables $x_{v_i,u}$ and $x_{v_i,w}$, the current linear system contradicts the clause $\neg x_{(v,u)} \vee \neg x_{(v,w)}$. A leaf ℓ in a branch that substitutes 1 to only one of variables $x_{(v_i,u)}$ is degenerate since $\Phi^{\ell}_{T_{u_i}}$ contradicts $\sum_{(v_i,u)\in E} x_{(v_i,u)} = 0$. And a leaf in a branch that substitutes 0 to all variables $x_{(v_i,u)}$ contradicts $\bigvee_{(v_i,u)\in E} x_{(v_i,u)}$. The size of T_{u_i} is at most $O(|V|^2)$ since in every leaf there are most two variables $x_{(v_i,u)}$ with value 1. Hence the size of T is at most $O(|V|^3)$. □

4 Lower bound for 2-fold Tseitin formulas

In this section we prove a lower bound on the size of a linear splitting tree. The proof consists of two parts. At first we transform a splitting tree to a communication protocol and then we prove a lower bound on the communication complexity.

Communication protocol from linear splitting tree. Let ϕ be an unsatisfiable CNF formula. For every assignment of its variables there exists a clause of ϕ that is falsified by the assignment. By $Search_\phi$ we denote a search problem where the instances are variables assignments of and the solutions are clauses of ϕ that are falsified by the assignment.

Let's consider some function or search a problem f with inputs $\{0,1\}^n$; the set $[n]$ is split into two disjoint sets X and Y. Alice knows bits of input corresponding to X and Bob knows bits of inputs corresponding to Y. A randomized communication protocol with public random bits and error ϵ is a binary tree such that every internal node v is labeled with a function of one of the two types: $a_v : \{0,1\}^X \times \{0,1\}^R \to \{0,1\}$ or $b_v : \{0,1\}^Y \times \{0,1\}^R \to \{0,1\}$, where R is an integer that denotes the number of random bits used by a protocol. For every internal node one of the edges to children is labeled with 0 and the other with 1, and leaves are labeled with strings (answers of a protocol). Assume that Alice knows $x \in \{0,1\}^X$ and Bob knows $y \in \{0,1\}^Y$; both of them know a random string $r \in \{0,1\}^R$. Alice and Bob communicate according to the protocol in the following way: initially they put a token in the root of the tree. Every time if the node with the token is labeled by a function of type a_v, then Alice computes the value of $a_v(x, r)$ and sends the result to Bob; and if the node is labeled by a function of type b_v, then Bob computes the value of $b_v(x, r)$ and sends the result to Alice. After this, both players move the token to the child that corresponds to the sent bit. The communication stops whenever the token moves to a leaf. The label in the leaf is the result of the communication with a given string of random bits r. It is required that with probability at least $1 - \epsilon$ over random choice of the string $r \leftarrow \{0,1\}^R$ the result of the protocol is a correct answer to the problem f. The complexity of a communication protocol is a depth of the tree or, equivalently, the number of bits that Alice and Bob must send in the worst case. By a randomized communication complexity with error ϵ of the problem f we call

a number $R^{pub}_{\epsilon}(f)$ that equals the minimal complexity of a protocol that solves f. See [11] for more details.

Let $EQ : \{0,1\}^{2n} \to \{0,1\}$, and for all $x, y \in \{0,1\}^n$, $EQ(x,y) = 1$ iff $x = y$. When we study the communication complexity of EQ we assume that Alice knows x and Bob knows y.

Lemma 4.1 ([11])**.** $R^{pub}_{\delta}(EQ) \leq \lceil \log \frac{1}{\delta} \rceil + 1$.

Proof. We consider the following protocol of depth 2: Alice sends the inner product $<x, r> = \sum_{i=1}^{n} x_i r_i \bmod 2$, where $r \in \{0,1\}^n$ is a random string. Bob computes the inner product $<y, r>$ and sends 1 if his result equals to the result of Alice and sends 0 otherwise. The result of this protocol equals to the bit sent by Bob. If $x = y$ then the result of the protocol is correct with probability 1. If $x \neq y$ then by the random subsum principle the result is correct with probability 1/2. In order to reduce the probability of error to δ, Alice sequentially sends $\lceil \log \frac{1}{\delta} \rceil$ inner products of x with independent random strings $r_1, r_2, \ldots, r_{\lceil \log \frac{1}{\delta} \rceil} \in \{0,1\}^n$. Then Bob verifies that his inner products $<y, r_1>, <y, r_2>, \ldots, <y, r_{\lceil \log \frac{1}{\delta} \rceil}>$ equal to inner products of Alice and sends 1 if everything is the same and 0 otherwise. If $x = y$ then the result of the protocol is correct with probability 1. If $x \neq y$, then by the random subsum principle the result is correct with probability $1 - \frac{1}{2^{\lceil \log \frac{1}{\delta} \rceil}} \geq 1 - \delta$. □

Lemma 4.2. Consider several equalities over $\mathbb{F}_2$. Let Alice knows values of some variables and Bob knows values of the other variables. There exists randomized public coin communication protocol with error δ that uses $O(\log \frac{1}{\delta})$ bits of communication.

Proof. Assume that we have to verify t equalities. When we verify the j-th equality, Alice have to compute the sum of her variables and Bob computes the sum of his variable. And we should verify that the sum of the results of Alice and Bob equals the right hand side of the equality α_j. Let the sum of Alice variables of the j-th equality plus α_i equals z_j and the sum of variables of Bob of the j-th equality equals y_j. All equalities are satisfied by π iff $EQ(z_1 z_2 \ldots z_t, y_1 y_2 \ldots y_t) = 1$. In order to compute EQ we use a protocol for EQ from the Lemma 4.1. □

Theorem 4.1. Let ϕ be an unsatisfiable CNF formula and T be a linear splitting tree for ϕ. Then for every distribution of variables of ϕ between Alice and Bob, $R^{pub}_{1/3}(Search_{\phi}) = O(\log |T| \log \log |T|)$.

Proof. We construct a communication protocol from the tree T without degenerate leaves. Alice and Bob together know an assignment π of variables of ϕ (Alice knows some bits of π and Bob knows the other bits of π). The assignment π determines a path ℓ_π in T that corresponds to edges with labels that are satisfied by π. This path contains a leaf that refutes some clause C_π of ϕ. The protocol that we are describing with high probability returns the clause C_π.

The protocol has $O(\log |T|)$ randomized rounds. In the analisys of the next round we will assume that all previous rounds do not contain errors. Thus the total error may be estimated as a sum of errors of the individual rounds. Both Alice and Bob at the beginning of the i-th round know a tree T_i that is a connected subgraph of T; $T_1 = T$. Since T_i is a connected subgraph of T, we may assume that the root of T_i is its highest

vertex in T. Under the assumption that all previous rounds were correct we will ensure that T_i contains the part of the path ℓ_π that goes from the root of T_i to the leaf that refutes C_π. We also maintain inequality $|T_{i+1}| \leq \frac{2}{3}|T_i|$. Thus if T_i has only one vertex it would be the leaf of T that refutes C_π, therefore Alice and Bob will know C_π.

Let $|T_i| > 1$, then there exists such a vertex v of T_i that the size of the subtree of T_i with the root v (we denote it by $T_i^{(v)}$) is at least $\frac{1}{3}|T_i|$ and at most $\frac{2}{3}|T_i|$. The tree T_{i+1} equals $T_i^{(v)}$ if v belongs to the path ℓ_π and equals $T_i \setminus T_i^{(v)}$ otherwise. Alice and Bob, using a fixed algorithm, find the vertex v; now they have to verify whether v belongs to ℓ_π. The vertex v belongs to the path ℓ_π iff π satisfies all equalities that are written along the path from the root of T_i to v. Alice and Bob verifies this equalities using Lemma 4.2 with $\delta = \frac{1}{3\lceil \log_{3/2}|T| \rceil}$. Since the number of rounds is at most $\lceil \log_{3/2}|T| \rceil$, the total error of the protocol is at most $\frac{1}{3}$. The total depth of the protocol is at most the number of rounds $\lceil \log_{3/2}|T| \rceil$ times the depth of the EQ protocol $O(\log\log|T|)$. □

Lower bound on communication complexity. A Tseitin formula $TS_{G,c}$ can be constructed from an arbitrary graph $G(V,E)$ and a function $c : V \to \mathbb{F}_2$; variables of $TS_{G,c}$ correspond to edges of G. The formula $TS_{G,c}$ is a conjunction of the following conditions encoded in CNF for every vertex v: the parity of the number of edges incident to v that have value 1 is the same as the parity of $c(v)$. It is well known that $TS_{G,c}$ is unsatisfiable if and only if $\sum_{v\in V} c(v) = 1$.

A k-fold Tseitin formula $TS^k_{(G,c)}$ [2] can be obtained from Tseitin formula $TS_{G,c}$ if we substitute every variable x_i by a conjunction of k new variables $(z_{i1} \wedge z_{i2} \wedge \cdots \wedge z_{ik})$ and translate the resulting formula into CNF. Note that if the maximal degree of G is bounded by a constant, then for every constant k the formula $TS^k_{(G,c)}$ has CNF representation of size polynomial in $|V|$.

Theorem 4.2. In time polynomial in n one may construct a graph $G(V,E)$ on n vertices with maximal degree bounded by a constant and a function $c : V \to \mathbb{F}_2$ such that $TS^2_{(G,c)}$ is unsatisfiable and $R^{pub}_{1/3}(Search_{TS^2_{(G,c)}}) = \Omega\left(\frac{n^{1/3}}{(\log(n)\log\log(n))^2}\right)$.

Corollary 4.1. In the condition of the Theorem 4.2 the size of any linear splitting tree of $TS^2_{(G,c)}$ is at least $\Omega\left(2^{n^{1/3}/\log^3(n)}\right)$.

Proof of Corollary 4.1. Follows from Theorem 4.2 and Theorem 4.1. □

We define a function $DISJ_{n,2} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that for all $x, y \in \{0,1\}^n$ $DISJ_{n,2}(x,y) = 1$ iff $x_i \wedge y_i = 0$ for all $i \in [n]$.

Theorem 4.3. ([2], Section 5) Let $m = \frac{n^{1/3}}{\log(n)}$, then in time polynomial in n one may construct a graph $G(V,E)$ on n vertices with maximal degree bounded by a constant and a function $c : V \to \mathbb{F}_2$ such that $TS^2_{(G,c)}$ is unsatisfiable and $R^{pub}_{\epsilon}(DISJ_{m,2}) = O\left(R^{pub}_{\epsilon}(Search_{TS^2_{(G,c)}})\log(n)(\log\log(n))^2\right)$.

Lemma 4.3. ([10]) $R^{pub}_{1/3}(DISJ_{n,2}) = \Omega(n)$.

Proof of Theorem 4.2. Let $m = \frac{n^{1/3}}{\log(n)}$. By Lemma 4.3, $R^{pub}_{\epsilon}(DISJ_{m,2}) = \Omega(m)$, then by theorem 4.3 it is possible to construct G and c such that $R^{pub}_{1/3}\left(Search_{TS^2_{(G,c)}}\right) = \Omega\left(\frac{n^{1/3}}{(\log(n)\log\log(n))^2}\right)$. □

5 Lower bound for Pigeonhole Principle

In this section we prove a lower bound on the size of linear splitting trees for formulas PHP^m_n that encode the pigeonhole principle. Formula PHP^m_n has variables $p_{i,j}$, where $i \in [m]$, $j \in [n]$; $p_{i,j}$ states that i-th pigeon is in the j-th hole. A formula has the two types of clauses: 1) Long clauses that encode that every pigeon is in some hole: $p_{i,1} \vee p_{i,2} \cdots \vee p_{i,n}$ for all $i \in [m]$; 2) Short clauses that encode that every hole contains at most one pigeon: $\neg p_{i,k} \vee \neg p_{j,k}$ for all $i \neq j \in [m]$ and all $k \in [n]$. If $m > n$ then PHP^m_n is unsatisfiable.

We call an assignment of values of variables $p_{i,j}$ *acceptable* if it satisfies all short clauses. In other words in every acceptable assignment there are no holes with two or more pigeons.

Lemma 5.1. Let a linear system $Ap = b$ from variables $p = (p_{i,j})_{i\in[m],j\in[n]}$ have at most $\frac{n-1}{2}$ equations and let it have an acceptable solution. Then for every $i \in [m]$ this system has an acceptable solution that satisfies the long clause $p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n}$.

Proof. Note that if we change 1 to 0 in an acceptable assignment, then it remains acceptable. Let the system have k equations; we know that $k \leq \frac{n-1}{2}$. We consider an acceptable solution π of the system $Ap = b$ with the minimum number of ones. We prove that the number of ones in π is at most k. Let the number of ones is greater than k. Consider $k+1$ variables that take value 1 in π: $p_{j_1}, p_{j_2}, \ldots, p_{j_{k+1}}$. Since the matrix A has k rows, the columns that correspond to variables $p_{j_1}, p_{j_2}, \ldots, p_{j_{k+1}}$ are linearly depended. Therefore there exists a nontrivial solution π' of the homogeneous system $Ap = 0$ such that every variable with value one in π' is from the set $\{p_{j_1}, p_{j_2}, \ldots, p_{j_{k+1}}\}$. The assignment $\pi' + \pi$ is also a solution of $Ap = b$ and is acceptable because $\pi' + \pi$ can be obtained from π by changing ones to zeros. Since π' is nontrivial, the number of ones in $\pi' + \pi$ is less then the number of ones in π and this contradicts the minimality of π.

The fact that π has at most k ones implies that π has at least $n-k$ empty holes. From the statement of the lemma we know that $n-k \geq k+1$; we choose $k+1$ empty holes with numbers $\ell_1, \ell_2, \ldots, \ell_{k+1}$. We fix $i \in [m]$; the columns of A that correspond to variables $p_{i,\ell_1}, \ldots, p_{i,\ell_{k+1}}$ are linearly depended, therefore there exists a nontrivial solution τ of the system $Ap = 0$ such that every variable with value 1 in τ is from the set $\{p_{i,\ell_1}, \ldots, p_{i,\ell_{k+1}}\}$. The assignment $\pi + \tau$ is a solution of $Ap = b$; $\pi + \tau$ is acceptable since holes with numbers $\ell_1, \ell_2, \ldots, \ell_{k+1}$ are empty in π, and τ puts at most one pigeon to them (if τ puts a pigeon in a hole, then this is the i-th pigeon). The assignment $\pi + \tau$ satisfies $p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n}$ because τ is nontrivial. □

Theorem 5.1. For all $m > n$ every linear splitting tree for PHP^m_n has size at least $2^{\frac{n-1}{2}}$.

Proof. We say that the equality $f = \alpha$ is acceptably implied from a linear system Φ if every acceptable solution of Φ satisfies $f = \alpha$.

We consider a linear splitting tree T for PHP_n^m. Remove from T all the vertices v for which Φ_v^T has no acceptable solutions. The resulting graph is a tree since if we remove a vertex, then we should remove its subtree, and the root of T is not removed. We denote this tree by T'. Note that it is impossible that a leaf of T' is not a leaf in T. Indeed, assume that v is labeled in T by a linear form f, then every acceptable assignment that satisfies Φ_v^T also satisfies one of the systems $\Phi_v^T \wedge (f = 1)$ or $\Phi_v^T \wedge (f = 0)$, so one of the children is not removed. Hence in every leaf ℓ of T' the system Φ_ℓ^T refutes a clause of PHP_n^m. Since there exists an acceptable assignment that satisfies Φ_ℓ^T, then Φ_ℓ^T can't refute short clause, therefore it refutes a long clause.

Consider a vertex v of T' with the only child u, let the edge (u, v) be labeled by $f = \alpha$. We know that the system $\Phi_v^T \wedge (f = 1 + \alpha)$ has no acceptable solutions. Hence the equality $f = \alpha$ is acceptably implied from Φ_v^T; and the sets of acceptable solutions of $\Phi_u^{T'}$ and $\Phi_v^{T'}$ are equal.

Let T' contain a vertex v with the only child u; we merge u and v in one vertex and remove the edge (u, v) with its label. We repeat this operation while the current tree has vertices with the only child. We denote the resulting tree by T''. Let V' be the set of vertices of T', and V'' be the set of vertices of T''. We define a surjective mapping $\mu : V' \to V''$ that maps a vertex from T' to a vertex of T'' into which it was merged. We know that for all $u \in T'$ the sets of acceptable solutions of $\Phi_u^{T'}$ and $\Phi_{\mu(u)}^{T''}$ are equal.

For every leaf ℓ'' of T'' there exists a leaf ℓ' of T' such that $\mu(\ell') = \ell''$, the system $\Phi_{\ell'}^{T'}$ refutes some long clause $p_{i,1} \vee \cdots \vee p_{i,n}$, therefore the system $\Phi_{\ell''}^{T''}$ has no acceptable solutions that satisfy $p_{i,1} \vee \cdots \vee p_{i,n}$. By construction all internal nodes of T'' have two children. Lemma 5.1 implies that the depth of all leaves in T'' is at least $\frac{n-1}{2}$, hence the size of T'' is at least $2^{(n-1)/2}$. □

6 Proof systems Res-Lin and Sem-Lin

A linear clause is a disjunction of linear equalities $\bigvee_{i=1}^k (f_i = \alpha_i)$, where f_i is a linear form and $\alpha_i \in \mathbb{F}_2$. Equivalently we may rewrite a linear clause as a negation of a system of linear equalities $\neg \bigwedge_{i=1}^n (f_i = 1 + \alpha_i)$. A trivial linear clause is a linear clause that is identically true. A clause $\neg \bigwedge_{i=1}^n (f_i = \alpha_i)$ is trivial iff the system $\bigwedge_{i=1}^n (f_i = \alpha_i)$ has no solutions.

A linear CNF formula is a conjunction of linear clauses. We say that propositional formula ϕ is semantically implied form the set of formulas $\psi_1, \psi_2, \ldots, \psi_k$ if every assignment that satisfies ψ_i for all $i \in [k]$ also satisfies ϕ.

We define a proof system Res-Lin that can be used to prove that a linear CNF formula is unsatisfiable. This system has two rules: 1)The weakening rule allows to derive from a linear clause C any linear clause D such that C semantically implies D. 2)The resolution rule allows to derive from linear clauses $(f = 0) \vee D$ and $(f = 1) \vee D'$ the linear clause $D \vee D'$.

A derivation of a linear clause C from a linear CNF ϕ in the Res-Lin system is a sequence of linear clauses that ends with C and every clause is either a clause of ϕ or it may be obtained from previous clauses by a derivation rule. The proof of the unsatisfiability of a linear CNF is a derivation of the empty clause (contradiction). The Sem-Lin system differs from Res-Lin by the second rule. It is replaced by a semantic rule

that allows to derive from linear clauses C_1, C_2 any linear clause C_0 such that C_1 and C_2 semantically imply C_0.

In order to verify that systems Sem-Lin and Res-Lin are proof systems in the sence of [4] we have to ensure that it is possible to verify a correctness of a proof in polynomial time. It is enough to verify a correctness of applications of rules. The correctness of the resolution rule is easy to verify, and for the verification of the other rules we use the following proposition.

Proposition 6.1. It is possible to verify in polynomial time: 1) whether a linear clause $C_0 = \neg \bigwedge_{i\in I}(f_i = \alpha_i)$ is a result of the weakening rule of $C_1 := \neg \bigwedge_{i\in J}(g_i = \beta_i)$; 2) whether a linear clause $C_0 := \neg \bigwedge_{i\in J}(g_i = \beta_i)$ is semantically implied from $C_1 := \neg \bigwedge_{i\in J}(g_i = \beta_i)$ and $C_2 = \neg \bigwedge_{i\in K}(h_i = \gamma_i)$.

Proof. 1) A linear clause C_0 is a weakening of C_1 iff any satisfying assignment of C_1 satisfies C_0. We show that C_0 is a weakening of C_1 iff for all $j \in J$ the system $\bigwedge_{i\in I}(f_i = \alpha_i) \wedge (h_j = \beta_j + 1)$ has no solutions. Indeed, if this system has a solution, then the solution satisfies C_1 and refutes C_0. Let C_0 be not a weakening of C_1, then there exists an assignment that satisfies C_1 and refutes C_0, this assignment satisfies an equality $h_j = \beta_j + 1$ for some $j \in J$, hence this assignment satisfies $\bigwedge_{i\in I}(f_i = \alpha_i) \wedge (h_j = \beta_j + 1)$.

Thus to verify a correctness of a weakening rule it is enough to check that for all $j \in J$ the corresponding system has no solution.

2) Similarly to item 1) it may be shown that C_0 is a semantic implication of C_1 and C_2 iff for all $j \in J$ and $k \in K$ the system $\bigwedge_{i\in I}(f_i = \alpha_i) \wedge (g_j = \beta_j + 1) \wedge (h_k = \gamma_k + 1)$ has no solution. □

Proposition 6.2. The weakening rule may be simulated by a polynomial number of applications of the following pure syntactic rules: 1) *The simplification* rule that allows to derive D from $D \vee (0 = 1)$; 2) *The syntactic weakening* rule that allows to derive $D \vee (f = \alpha)$ from D; 3) *The addition* rule that allows to derive $D \vee (f_1 = \alpha_1) \vee (f_1 + f_2 = \alpha_1 + \alpha_2 + 1)$ from $D \vee (f_1 = \alpha_1) \vee (f_2 = \alpha_2)$.

Proof. It is more convenient to represent a linear clause as the negation of a linear system. In this representation the addition rule allows to add one from the system equality to another and the simplification rule is just a removing the trivial equality $0 = 0$.

Let a clause $\neg \bigwedge_{i\in I}(g_i = \beta_i)$ is the result of the weakening rule applied to $\neg \bigwedge_{i\in J}(f_i = \alpha_i)$.

At first we apply multiple syntactic weakening and get $\neg \bigwedge_{i\in J}(f_i = \alpha_i) \wedge \bigwedge_{i\in I}(g_i = \beta_i)$.

From the proof of Proposition 6.1 we know that every equality $f_i = \alpha_i$ is a linear combination of equalities $g_j = \beta_j$. Thus we may get $0 = 0$ from every $f_i = \alpha_i$ by multiple application of the addition rule. And finally we remove all $0 = 0$ by the simplification rule.

□

We show that systems Sem-Lin and Res-Lin are polynomially equivalent. It means that any proof in one system may be translated to the proof in other system in polynomial time. Every proof in Res-Lin is also a proof in Sem-Lin; the next proposition is about the opposite translation.

Proposition 6.3. Let nontrivial linear clause $C_0 := \neg\{f_i = \alpha_i\}_{i\in I}$ be a semantic implication of $C_1 := \neg\{g_i = \beta_i\}_{i\in J}$ and $C_2 := \neg\{h_i = \gamma_i\}_{i\in L}$. Then C_0 can be obtained from C_1 and C_2 by applications of at most one resolution rule and several weakening rules.

Before we start a proof we consider an example that shows how the linear clause $(x+y=0)$ can be derived from $(x=0)$ and $(y=0)$ in Res-Lin: 1) Apply weakening rule to $(x=0)$ and get $(x+y=0)\vee(y=1)$; 2) Apply resolution rule to $(x+y=0)\vee(y=1)$ and $(y=0)$ and get $(x+y=0)$.

We will use the following well known lemma:

Lemma 6.1. If for a matrix $A \in \mathbb{F}_2^{m\times n}$ and a vector $b \in \mathbb{F}_2^m$ the linear system $Ax = b$ has no solutions, then there exists a vector $y \in \mathbb{F}_2^m$ such that $y^T A = 0$ and $y^T b = 1$. In other words if a linear system over $\mathbb{F}_2$ is unsatisfiable then it is possible to sum several equations and get a contradiction $0 = 1$.

Proof of Proposition 6.3. Both C_1 and C_2 can't be trivial since in this case C_0 must be trivial. If C_i for $i \in \{1, 2\}$ is trivial, then C_0 is a weakening of C_{2-i}. So we assume that C_1 and C_2 are not trivial.

For all $j \in J$ and $l \in L$ the system $\bigwedge_{i\in I}(f_i = \alpha_i) \wedge (g_j = 1 + \beta_j) \wedge (h_l = 1 + \gamma_l)$ is unsatisfiable. Since the system $\bigwedge_{i\in I}(f_i = \alpha_i)$ is satisfiable, one of the following holds: 1)$\bigwedge_{i\in I}(f_i = \alpha_i)$ becomes unsatisfiable if we add just one equality (for example $g_j = 1+\beta_j$). Then by Lemma 6.1 the negation of this equality can be obtained as a linear combination of equalities from $\bigwedge_{i\in I}(f_i = \alpha_i)$. 2) The system $\bigwedge_{i\in I}(f_i = \alpha_i)$ becomes unsatisfiable only if we add both equalities $(g_j = 1 + \beta_j) \wedge (h_l = 1 + \gamma_l)$. By Lemma 6.1 the equality $g_j+h_l = \beta_j+\gamma_l+1$ may be obtained as a linear combination of equalities from the system $\bigwedge_{i\in I}(f_i = \alpha_i)$. Note that if equalities $g_j = 1 + \beta_j$ and $h_l = 1 + \gamma_l$ contradict each other (i.e. $g_j = h_l$ and $\beta_j = 1 + \gamma_l$), then the equality $g_j + h_l = \beta_j + \gamma_l + 1$ is just $0 = 0$.

We split J into two disjoint sets J' and J'', where $j \in J''$ iff the system $\bigwedge_{i\in I}(f_i = \alpha_i) \wedge (g_j = \beta_j + 1)$ is unsatisfiable. Similarly we define a splitting $L = L' \cup L''$. Note that if $J = J''$, then $\neg\bigwedge_{i\in I}(f_i = \alpha_i)$ is a weakening of $\neg\bigwedge_{j\in J}(g_j = \beta_j)$, similarly if $L = L''$, then $\neg\bigwedge_{i\in I}(f_i = \alpha_i)$ is a weakening of $\neg\bigwedge_{i\in L}(h_i = \gamma_i)$. Thus in what follows we assume that $J' \neq \emptyset$ and $L' \neq \emptyset$.

We get that C_0 is a weakening of $D := \neg(\bigwedge_{i\in J''}(g_i = \beta_i) \wedge \bigwedge_{i\in L''}(h_i = \gamma_i) \wedge \bigwedge_{i\in J',j\in L'}(g_i + h_j = \beta_i + \gamma_j + 1)$. It remains to show that D can be obtained from C_1 and C_2 by application of one resolution rule and several weakening rules.

Let $j_0 \in J'$ and $l_0 \in L'$.

1) Apply the weakening rule to C_1 and get $D_1 := \neg\left((g_{j_0} = \beta_{j_0}) \wedge \bigwedge_{i\in J'}(g_i + h_{l_0} = \beta_i + \gamma_{l_0} + 1) \wedge \bigwedge_{i\in J''}(g_i = \beta_i)\right)$;

2) Apply the weakening rule to C_2 and get $D_2 := \neg\left((g_{j_0} = \beta_{j_0} + 1) \wedge \bigwedge_{i\in L'}(h_i + g_{j_0} = \beta_{j_0} + \gamma_{l_0} + 1) \wedge \bigwedge_{i\in L''}(h_i = \gamma_i\})\right)$;

3) Apply the resolution rule to D_1 and D_2, and get

$$D_3 := \neg\left(\bigwedge_{i\in J'}(g_i + h_{l_0} = \beta_i + \gamma_{l_0} + 1) \wedge \bigwedge_{i\in L'}(h_i + g_{j_0} = \beta_{j_0} + \gamma_{l_0} + 1) \wedge \bigwedge_{i\in J''}(g_i = \beta_i) \wedge \bigwedge_{i\in L''}(h_i = \gamma_i)\right)$$

4) Apply the weakening rule to D_3 and get D. □

6.1 Tree-like Res-Lin and linear splitting trees.

A proof in Res-Lin (or Sem-Lin) is tree-like if all clauses can be put in the nodes of a rooted tree in such a way that 1) the empty clause is in the root; 2) the clauses of an initial formula are in the leaves; 3) a clause in every internal node is a result of a rule of its children.

Linear splitting trees are naturally generalized to linear CNFs.

Lemma 6.2. 1) Every linear splitting tree for an unsatisfiable linear CNF may be translated into a tree-like Res-Lin proof and the size of the resulting proof is at most twice the size of the splitting tree. 2) Every tree-like Res-Lin proof of an unsatisfiable formula ϕ may be translated to a linear splitting tree for ϕ without increasing the size of the tree.

Proof. 1) We start from translation of a splitting tree to a splitting tree without degenerate leaves described in Proposition 2.1. We denote the resulting tree by T. On every vertex v of T we put a linear clause $\neg\Phi_v^T$. By construction every clause is a result of the resolution rule of clauses in its children; the root contains the empty clause. In every leaf ℓ the system Φ_ℓ^T refutes some linear clause C of the initial formula. Hence $\neg T_v$ is a weakening of C. The size of the resulting proof exeeds the size of the tree T only beacause of weakening rules in the leaves.

2) Consider a splitting tree and contract all edges that correspond to the weakening rule. We denote the resulting tree by T. All other edges correspond to applications of resolution rules. Let the resolution rule be applied to clauses $\neg((f=0)\wedge D_1)$ and $\neg((f=1)\wedge D_2)$, then we label an edge to the first of them by $f=0$ and to the second by $f=1$.

We show that for every vertex v all clauses written in v contradict to the system Φ_v^T. Since every vertex may contain several clauses (since we merge weakening rules) it is enough to prove this for the weakest clause in the vertex (i.e., to the clause that is a premise of the resolution rule). The proof is by induction on the depth of the vertex v. The root contains contradictory clause, hence the statement is true for the root. Assume that we prove a statement for vertex v, now we prove it for its children u and w. Let $\neg(D_1\wedge D_2)$ be a clause in v and let it be a result of the resolution rule applied to $\neg(f=0\wedge D_1)$ and $\neg(f=1\wedge D_2)$. By the induction hypothesis we know that $\neg(D_1\wedge D_2)$ contradicts the system Φ_v^T. It means that the negation of every equality in D_1 contradicts to Φ_v^T. Let $\neg(f=0\wedge D_1)$ be in the vertex u, then $\Phi_u^T=\Phi_v^T\wedge(f=0)$, hence $f=1$ contradicts Φ_v^T; negations of all equalities from D_1 contradict Φ_v^T and therefore contradict Φ_u^T. So we get that Φ_u^T contradicts $\neg(f=0\wedge D_1)$ and the similar is true for $\neg(f=1\wedge D_2)$. Applying the statement to leaves we get that every leaf refutes a clause of formula ϕ. □

Corollary 6.1. 1) For all $m>n$ every tree-like proof in Res-Lin and Sem-Lin of PHP_n^m has size $2^{\Omega(n)}$. 2) In the conditions of Theorem 4.2 the size of any tree-like resolution proof in Res-Lin and Sem-Lin of $TS^2_{(G,c)}$ is at least $\Omega(2^{n^{\frac{1}{3}}/\log^3(n)})$.

Proof. Follows from Lemma 6.2, Proposition 6.3, Theorem 5.1 and Theorem 4.2. □

6.2 Implication completeness of Res-Lin

Now we prove that Res-Lin is implication complete. The following lemma is straightforward.

Lemma 6.3. 1) If a linear clause D is a weakening of a linear clause C, then for every linear clause E the clause $D \vee E$ is a weakening of $C \vee E$. 2) If a linear clause D is a semantic implication of (or a result of the resolution rule applied for) C and F, then for every linear clause E the clause $D \vee E$ is a semantic implication (or a result of the resolution rule applied for) $C \vee E$ and $F \vee E$.

Theorem 6.1. If a linear clause C_0 is a semantic implication of $C_1, C_2, \ldots, C_k$, then C_0 may be derived from $C_1, C_2, \ldots, C_k$ in Res-Lin.

Proof. The plan of the proof is following: we construct a list of linear clauses $\mathcal{D}$ such that the conjunction of clauses from $\mathcal{D}$ is unsatisfiable. Since Res-Lin is complete (Res-Lin is complete because every linear CNF has a splitting tree with splitting over all variables), then there exists a derivation of the empty clause from $\mathcal{D}$. By Lemma 6.3 from the list $\mathcal{D}' := \{D \vee C_0 \mid D \in \mathcal{D}\}$ it is possible to derive C_0. After this we show that every clause in $\mathcal{D}'$ is a weakening of some clause among $C_1, C_2, \ldots, C_k$.

We construct the list $\mathcal{D}$ step by step; initially $\mathcal{D}$ consists of clauses $C_1, C_2, \ldots, C_k$. Note that if an assignment π refutes C_0, then by the statement of the theorem it also refutes one of the clauses $C_1, C_2, \ldots, C_k$, hence it refutes their conjunction. Let $C_1 := \bigvee_{i=1}^{n}(f_i = \alpha_i)$ and $C_0 := \bigvee_{i=1}^{m}(g_i = \beta_i)$ While there exists such an assignment π that satisfies C_0 and satisfies all clauses from $\mathcal{D}$, we add to the list $\mathcal{D}$ a new clause C^π. Since π satisfies C_0, then there exists i such that π satisfies $g_i = \beta_i$. Let's denote $I := \{i \mid \pi \text{ satisfies} f_i = \alpha_i\}$ and let the clause C^π equal $\bigvee_{i \in I}(f_i + g_i = \alpha_i + \beta_i + 1) \vee \bigvee_{i \notin I}(f_i = \alpha_i)$. By construction π refutes C^π.

Finally, for every assignment of variables there exists such a clause in the list $\mathcal{D}$ that is not satisfied by the assignment. Hence the conjunction of clauses from $\mathcal{D}$ is unsatisfiable. We have to show that for all $D \in \mathcal{D}$ the clause $D \vee C_0$ is a weakening of some clause among $C_1, C_2, \ldots, C_k$. If D equals one clause from $C_1, C_2, \ldots, C_k$, we are done. Let $D = C^\pi$, then $D \vee C_0$ is a weakening of $C_1 \vee C_0$ and therefore is a weakening of C_1. □

6.3 Simulation of Res-Lin in $R(lin)$

In this section we show that the system $R(lin)$ p-simulates Res-lin. The system $R(lin)$ operates with linear equalities over integer coefficients and propositional variables. In this section we use sign $=$ for equality of integers and sign $\underset{2}{=}$ for equality modulo 2. An integer linear clause is the disjunction $\bigvee_i(\sum_j a_{i,j}x_j = b_i)$, where $a_{i,j}$ and b_j are integers. Equalities in a clause are not repeated.

The proof system $R(lin)$ contains axioms $(x = 0) \vee (x = 1)$ for all variables x and the following inference rules:

- The cut rule that allows to deduce clauses $A \vee B \vee (F_1 + F_2 = a_1 + a_2)$ and $A \vee B \vee (F_1 - F_2 = a_1 - a_2)$ from $A \vee (F_1 = a_1)$ and $B \vee (F_2 = a_2)$.
- The syntactic weakening that allows to deduce $A \vee (F = a)$ from A for every integer linear equality $F = a$.
- The simplification rule that allows to deduce B from $B \vee (0 = c)$, where c is nonzero integer.

By means of $R(lin)$ one may prove that a set of integer linear clauses $K = \{K_1, \ldots, K_m\}$ is contradictory. Namely a proof is a sequence of integer linear clauses that ends with empty clause and every clause in this sequence is either an axiom or a clause form K or may be obtained from previous clauses by application of an inference rule.

An equality $x_1 + x_2 + \cdots + x_n \underset{2}{\equiv} 0$ is represented by the following disjunction of integer linear equalities: $(x_1 + x_2 + \cdots + x_n = 0) \vee (x_1 + x_2 + \cdots + x_n = 2) \vee \cdots \vee (x_1 + x_2 + \cdots + x_n = 2\lceil \frac{n}{2} \rceil)$ and an equality $x_1 + x_2 + \cdots + x_n \underset{2}{\equiv} 1$ is represented by the following disjunction of integer linear equalities: $(x_1 + x_2 + \cdots + x_n = 1) \vee (x_1 + x_2 + \cdots + x_n = 3) \vee \cdots \vee (x_1 + x_2 + \cdots + x_n = 2\lceil \frac{n-1}{2} \rceil + 1)$.

Theorem 6.2. The system $R(lin)$ p-simulates Res-Lin.

Proof. By Proposition 6.2 it is enough to p-simulate the resolution rule, the simplification rule, the syntactic weakening and the addition rule in $R(lin)$. The simulation of the simplification rule and the syntactic weakening rule are straightforward. Thus we have to p-simulate the resolution rule and the addition rule.

Lemma 6.4. It is possible to deduce $A \vee B \vee \bigvee_{i\in[k],j\in[n]}(L_i \pm K_j)$ from $A \vee \bigvee_{i=1}^{k} L_i$ and $B \vee \bigvee_{j=1}^{n} K_j$ in $R(lin)$, where L_i and K_j are equalities with integer coefficients. The statement holds for all variants of signs $\pm$.

Proof. We denote $C := \bigvee_{i\in[k],j\in[n]}(L_i \pm K_j)$.

We apply multiple syntactic weakening rules to $B \vee \bigvee_{j=1}^{n} K_j$ and get $A \vee B \vee C \vee \bigvee_{j=1}^{n} K_j$. Now we will successively eliminate extra equalities starting from the end.

Assume that we have a clause $A' := A \vee B \vee C \vee \bigvee_{j=1}^{\ell} K_j$, where $\ell \geq 1$.

We apply the cut rule to $A \vee \bigvee_{i=1}^{k} L_i$ and A' and get $B'' = A \vee B \vee C \vee \bigvee_{j=1}^{\ell-1} K_j \vee \bigvee_{i=1}^{k-1} L_i$, here the equality $L_k \pm K_\ell$ is contained in C, therefore we do not write it the second time.

Now we apply the cut rule to B'' and A' and we eliminate the last equality from B''. We apply the application of the cut rule several times and finally get $A'' = A \vee B \vee C \vee \bigvee_{j=1}^{\ell-1} K_j$, thus we reduce the number of equalities K_j with respect to A'. □

The resolution rule can be simulated by the application of Lemma 6.4 and simplification rules. Indeed to apply the resolution rule to $A \vee f = 1 \vee f = 3 \vee \ldots$ and $B \vee f = 0 \vee f = 2 \vee \ldots$ we apply Lemma 6.4 and get $A \vee B \vee 0 = 1 \vee 0 = 3 \vee \ldots$, and finally by application of simplification rules we get $A \vee B$.

Lemma 6.5. Let $f(x) = a_1x_1 + a_2x_2 + \cdots + a_nx_n$, where $a_1, a_2, \ldots, a_n$ are natural numbers, then $(f(x) = 0) \vee \cdots \vee (f(x) = \sum\limits_i a_i)$ is deducible in $R(lin)$.

Proof. We use a_i times Lemma 6.4 and get $(a_ix_i = 0) \vee (a_ix_i = 1) \vee \ldots (a_ix_i = a_i)$ from the axiom $(x_i = 0) \vee (x_i = 1)$. Then we apply Lemma 6.4 for all $i \in [n]$ and get the desired clause. □

The simulation of the addition rule in $R(lin)$ follows from the following lemma:

Lemma 6.6. It is possible to deduce $A \vee (f \underset{2}{\equiv} \alpha) \vee (f + g \underset{2}{\equiv} \alpha + \beta + 1)$ from $A \vee (f \underset{2}{\equiv} \alpha) \vee (g \underset{2}{\equiv} \beta)$ in polynomial steps in $R(lin)$, where $\alpha, \beta \in \mathbb{F}_2$ and $f = \sum_{i\in I} x_i$ and $g = \sum_{j\in J} x_j$ are linear forms.

Proof. We use Lemma 6.5 and get $(f \underset{2}{\equiv} \alpha) \vee (f \underset{2}{\equiv} \alpha + 1)$. Now we use Lemma 6.4 for this clause and the clause from the statement of the lemma and get $A \vee (f \underset{2}{\equiv} \alpha) \vee (f + g \underset{2}{\equiv} \alpha + \beta + 1)$. If sets I and J are disjoint then we are done.

Asume that $I \cap J \neq \emptyset$, then the equality $(f(x) + g(x) = 1 \mod 2)$ contains variables with coeffitient 2; we consider one such variable x_ℓ. From the axiom $(x_\ell = 0) \vee (x_\ell = 1)$ we deduce $(2x_\ell = 0) \vee (2x_\ell = 2)$ by two applications of the cut rule; and by Lemma 6.4 we get $D := A \vee (f \underset{2}{\equiv} \alpha) \vee (f + g - 2x_\ell \underset{2}{\equiv} \alpha + \beta + 1) \vee (f - g - 2x_\ell = -1)$. We have to eliminate the last equality in D. In order to do it we use Lemma 6.5 for linear form $f + g - 2x_\ell$ and get C. We apply the cut rule to C and D and repeat applying the cut rule to the result and D until we get $A \vee (f \underset{2}{\equiv} \alpha) \vee (f + g - 2x_\ell \underset{2}{\equiv} \alpha + \beta + 1)$. We have to repeat the same for other common variables of f and g. □

□

6.4 Space vs size tradeoff

We define the space complexity of Res-Lin proofs similarly to the Resolution. We assume that a proof is realized in the working memory. And there are the following basic operations: 1) To download a clause of the formula to the memory; 2) To remove a clause from the memory; 2) To deduce a clause form clauses in the memory using inference rules and add it to the memory. A clause space of a proof is the maximum number of clauses in the memory. We denote a clause space of π as $\mathrm{CSpace}(\pi)$ and the number of operations in π as $\mathrm{Size}(\pi)$

Remark 6.1. The protocol from Lemma 4.2 may be used to verify whether the linear clause is satisfied by a substitution of variables one part of that known by Alice and other part known by Bob.

Theorem 6.3. Let π be a Res-Lin proof of formula ϕ then $R^{pub}_{1/3}(Search_\phi) \leq O(\mathrm{CSpace}(\pi) \log \mathrm{Size}(\pi) \log(\mathrm{CSpace}(\pi) \log \mathrm{Size}(\pi)))$.

Proof. Let $S_0, S_1, S_2, \ldots, S_k$ be states of the memory of the proof π, $k = \mathrm{Size}(\pi)$. Alice and Bob using binary search will find $i \in [k]$ such that all clauses in S_{i-1} are satisfied and not all clauses from S_i are satisfied. If such i is found and there are no errors in the protocol, then ith operation is the uploading of a clause of ϕ that is not satisfied by the substitution and it would be the answer of the protocol.

By Remark 6.1 there is a protocol that verifies whether ℓ linear clauses are satisfied by the substitution with error $\ell\epsilon$ and $O(\ell \log \frac{1}{\epsilon})$ bit of communications.

The total error is at most $\mathrm{CSpace}(\pi)\epsilon \log \mathrm{Size}(\pi)$ and the number of bits of communication is at most $O(\mathrm{CSpace}(\pi) \log \frac{1}{\epsilon} \log \mathrm{Size}(\pi))$. Finally assume that $\epsilon = \frac{1}{3 \, \mathrm{CSpace}(\pi) \log \mathrm{Size}(\pi)}$.

Corollary 6.2. In the conditions of the Theorem 4.2 for all Res-Lin proof π of formula $TS^2_{(G,c)}$ the following holds: $\mathrm{CSpace}(\pi) \log \mathrm{Size}(\pi) \log(\mathrm{CSpace}(\pi) \log \mathrm{Size}(\pi)) \geq \Omega\left(\frac{n^{1/3}}{(\log(n) \log\log(n))^2}\right)$.

□

Acknowledgements. The authors are grateful to Jan Krajíček, Edward A. Hirsch and Alexander Knop for fruitful discussions. The authors also thanks Jan Krajíček for the statement of the problem, Alexander Shen for the suggestion to simplify the presentation of the first lower bound and to anonymous reviewers for multiple helpful comments.

References

[1] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *J. Autom. Reason.*, 35(1-3):51–72, 2005.

[2] Paul Beame, Toniann Pitassi, and Nathan Segerlind. Lower bounds for lovász-schrijver systems and beyond follow from multiparty communication complexity. *SIAM Journal on Computing*, 37(3):845–869, 2007.

[3] E. Ben-Sasson and A. Wigderson. Short proofs are narrow — resolution made simple. *Journal of ACM*, 48(2):149–169, 2001.

[4] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, March 1979.

[5] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[6] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[7] Evgeny Demenkov and Alexander S. Kulikov. An elementary proof of a 3n - o(n) lower bound on the circuit complexity of affine dispersers. In *MFCS*, pages 256–265, 2011.

[8] D. Itsykson and D. Sokolov. The complexity of inversion of explicit Goldreichs function by DPLL algorithms. In *Proceedings of CSR 2011*, volume 6651 of *Lecture Notes in Computer Science*, pages 134–147. Springer, 2011.

[9] Dmitry Itsykson. Lower bound on average-case complexity of inversion of goldreich's function by drunken backtracking algorithms. *Theory Comput. Syst.*, 54(2):261–276, 2014.

[10] Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, November 1992.

[11] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[12] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

[13] Alexander A. Razborov. Pseudorandom generators hard for k-dnf resolution and polynomial calculus resolution. Technical report, 2003.

[14] Alexander A. Razborov. Resolution lower bounds for perfect matching principles. *Journal of Computer and System Sciences*, 69(1):3–27, 2004.

[15] Kazuhisa Seto and Suguru Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Computational Complexity*, 22(2):245–274, 2013.

[16] G. S. Tseitin. On the complexity of derivation in the propositional calculus. *Zapiski nauchnykh seminarov LOMI*, 8:234–259, 1968. English translation of this volume: Consultants Bureau, N.Y., 1970, pp. 115–125.

[17] Alasdair Urquhart. The depth of resolution proofs. *Studia Logica*, 99(1-3):249–364, 2011.