On Limits of Symbolic Approach to SAT Solving

Dmitry Itsykson
Ben-Gurion University of the Negev, Beer-Sheva, Israel
On leave from Steklov Institute of Mathematics at St. Petersburg, Russia

Sergei Ovcharov
St. Petersburg State University, Russia

Abstract

We study the symbolic approach to the propositional satisfiability problem proposed by Aguirre and Vardi in 2001 based on OBDDs and symbolic quantifier elimination. We study the theoretical limitations of the most general version of this approach where it is allowed to dynamically change variable order in OBDD. We refer to algorithms based on this approach as $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms.

We prove the first exponential lower bound of $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms on unsatisfiable formulas, and give an example of formulas having short tree-like resolution proofs that are exponentially hard for $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms. We also present the first exponential lower bound for natural formulas with clear combinatorial meaning: every $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithm runs exponentially long on the binary pigeonhole principle BPHP_n^{n+1}.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Symbolic quantifier elimination, OBDD, lower bounds, tree-like resolution, proof complexity, error-correcting codes, binary pigeonhole principle

Digital Object Identifier 10.4230/LIPIcs.SAT.2024.19

Funding *Dmitry Itsykson*: Supported by European Research Council Grant No. 949707.

1 Introduction

The Boolean satisfiability problem (SAT) is the decision problem, where given a CNF formula we are to decide whether it is satisfiable or not. The symbolic approach to SAT is to gradually transform the input formula into a special representation model in which the satisfiability problem can be easily solved. In our case, the representation model is an ordered binary decision diagram (OBDD) due to Bryant [4]. An ordered binary decision diagram (OBDD) represents a Boolean function as a branching program with two sinks such that on every path from the source to a sink, variables appear in the same order. This restriction on the order of variables allows handling of the diagrams very efficiently.

Assume that the input formula is $\varphi = \bigwedge_{i=1}^{m} C_i$. The naive symbolic approach is to choose some permutation σ of the set $[m]$ and iteratively compute OBDD representing $\bigwedge_{i=1}^{k} C_{\sigma(i)}$ for $k = 1, 2, \ldots, m$. Algorithms implementing this approach vary in the way of choosing a permutation σ (for example, it can be chosen dynamically and not straightaway) and in the way of choosing the order of variables in OBDD. Algorithms that choose the one order of variables in OBDD and do not change it during the execution we denote by $\mathrm{OBDD}(\wedge)$ algorithms. Algorithms that dynamically change the order in OBDD we denote by $\mathrm{OBDD}(\wedge, \mathrm{reordering})$ algorithms.

Aguirre and Vardi suggested a smarter symbolic approach called symbolic quantifier elimination [1, 19]. The key idea is that we have to obtain OBDD representation not of the formula $\varphi(x_1, x_2, \ldots, x_n)$ itself but rather $\exists x_1 \exists x_2 \ldots \exists x_n \varphi$. On one hand, the exterior quantifiers are not necessary to be applied since the satisfiability testing is easy for OBDD. But in some cases, one can move the quantifier in the middle of the formula. Namely, if all occurrences of a variable x_j are among clauses $C_{\sigma(i)}$ for $i \in [k]$, then the quantifier for x_j

can be put in front of $\bigwedge_{i=1}^{k} C_{\sigma(i)}$ and we can apply projection over x_j to the current OBDD. In other words, now the current OBDD represents not a conjunction of clauses $\bigwedge_{i=1}^{k} C_{\sigma(i)}$ but rather $\exists x_{i_1} \dots \exists x_{i_\ell} \bigwedge_{i=1}^{k} C_{\sigma(i)}$, where the variables $x_{i_1}, \dots, x_{i_\ell}$ do not appear in $C_{\sigma(i)}$ for $i \in \{k+1, \dots, n\}$. Algorithms implementing this approach vary on how they can choose the permutation σ (possibly in a dynamic way), variables orders in OBDDs, and also on the way of moving existential quantifiers inside the formula if it is permitted. Similarly, we call such algorithms by OBDD($\wedge, \exists$) and OBDD($\wedge, \exists$, reordering) algorithms depending on whether it is allowed to change variable orders in OBDDs.

In this paper, we study theoretical lower bounds on the running time of OBDD($\wedge, \exists$, reordering) algorithms. The running time of every such algorithm can be bounded below by the maximal size of constructed OBDDs. Our goal is to construct hard formulas for which any algorithm must construct an OBDD of exponential size not dependently on choosing a permutation σ, variable orders in OBDDs, and moving of existential quantifiers.

Previous results. The study of OBDD algorithms is highly connected with the study of OBDD-based proof systems initiated by Atserias, Kolaitis, and Vardi [3] and then continued by other researchers [14, 6, 5]. Lower bounds for OBDD algorithms follow from the derivation size lower bounds in corresponding proof systems (if such lower bounds are known). In contrast with DPLL and CDCL algorithms [2, 17] proving lower bounds on satisfiable instances for OBDD algorithms is not harder than proving lower bound of unsatisfiable instances. Indeed, if we know that φ is a hard unsatisfiable formula for OBDD($\wedge, \dots$) algorithms then the formula $x \vee \varphi$ that is obtained from φ by addition of the new variable x to all clauses of φ is a hard satisfiable formula for the same class of OBDD($\wedge, \dots$) algorithms. However, we do not know such a reduction in the other direction.

Lower bounds for OBDD($\wedge$, reordering) algorithms on satisfiable formulas are rather easy to prove. Indeed, since at the end, the algorithm necessarily represents the initial formula, it is enough to construct a family of Boolean functions that have small CNF representations but require OBDD of exponential size for every order of variables. For example, satisfiable Tseitin formulas have such property [10]. Proving a lower bound for OBDD($\wedge$, reordering) algorithms on unsatisfiable formulas is harder since at the end of the execution the OBDD represents the constant false and hence it has a constant size. However, exponential lower bounds for OBDD($\wedge$, reordering) algorithms on unsatisfiable formulas are implied from the lower bound for corresponding OBDD based proof system; for example, unsatisfiable Tseitin formulas on expanders and the pigeonhole principle are hard for OBDD($\wedge$, reordering) algorithms [14].

At the same time unsatisfiable and satisfiable Tseitin formulas [14] and the pigeonhole principle [7, 18] are easy for OBDD($\wedge, \exists$) algorithms.

Lower bounds for running time of OBDD($\wedge, \exists$) algorithms on unsatisfiable formulas follow from the lower bounds of the derivation complexity in dag-like and tree-like OBDD based proof systems [15, 20]. Unfortunately, the mentioned lower bounds' proofs rely significantly on the fact that all OBDDs use the same variable order. One more disadvantage of these results is that the hard formulas are very artificial and they were constructed especially for the proof of lower bounds.

Buss, Itsykson, Knop, and Sokolov showed that the ability to change order in OBDDs makes OBDD proof systems stronger [6]. One can use the same technique to show that the same is true for OBDD algorithms.

The paper [14] gives an exponential lower bound on the running time of OBDD($\wedge, \exists$, reordering) algorithms on satisfiable formulas encoding that x is a codeword of the specific linear code. The question about lower bounds for OBDD($\wedge, \exists$, reordering) algorithms on unsatisfiable formulas was open before this paper.

Some restricted lower bounds follow from the result of the paper [5] for corresponding restricted proof systems. Namely, unsatisfiable Tseitin formulas are hard for $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms if we bound the number of quantifiers that can be moved inside the formula; recall that such formulas are easy without this restriction. Also, the paper [5] gives an example of unsatisfiable formulas that are hard for $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms if it is allowed to use only a small number (at most $c \log n$, where c is a small constant and n is the number of variables) of orders.

Our results. In this paper, we give two families of unsatisfiable formulas and we prove for them that they require exponential running time of $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms.

The first family is based on a combination of hard satisfiable formulas for $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms and hard unsatisfiable formulas for $\mathrm{OBDD}(\wedge, \mathrm{reordering})$ algorithms. The proof strategy is the following. The algorithm either does not apply projections and, thus, it simulates an $\mathrm{OBDD}(\wedge, \mathrm{reordering})$ algorithm on the hard unsatisfiable formula, or it applies projection and then it has to simulate the running of an $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithm on the hard satisfiable formula.

Using this approach we can get a stronger result. Namely, we construct hard formulas for $1{-}\mathrm{NBP}(\wedge, \exists)$ algorithms that use non-deterministic read-once branching programs ($1{-}\mathrm{NBP}$) instead of OBDDs as the base representation model. $1{-}\mathrm{NBP}$ extends OBDD and is strictly more efficient. We should stress that $1{-}\mathrm{NBP}(\wedge, \exists)$ algorithms have no practical sense since for $1{-}\mathrm{NBP}$ we cannot efficiently compute the result of the conjunction with a clause. But a lower bound on the size of $1{-}\mathrm{NBP}$ trivially implies the same lower bound on the size of OBDD. We apply this extension in Theorem 20 to get that $1{-}\mathrm{NBP}(\wedge, \exists)$ algorithms and, thus, $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms do not polynomially simulate tree-like resolution. This result extends the result of Segerlind that $\mathrm{OBDD}(\wedge, \exists)$ algorithms do not simulate dag-like resolution [20]. The separation formula can be obtained from the hard formula for $1{-}\mathrm{NBP}(\wedge, \exists)$ algorithms by adding several extension axioms. In Lemma 18 we show that adding extension axioms can not make the input formula simpler for $1{-}\mathrm{NBP}(\wedge, \exists)$ algorithm; the proof essentially exploits non-determinism in $1{-}\mathrm{NBPs}$. On the other hand, it is known that tree-like resolution with extension axioms is equivalent to the exceptionally strong proof system Extended Frege.

The disadvantage of the first family of hard formulas is that they are very artificial. So we prove the second lower bound for the formulas with a clear combinatorial meaning. The second family of hard formulas is the binary pigeonhole principle $\mathrm{BPHP}_{2^\ell}^{2^\ell+1}$ that encodes that there are $2^\ell + 1$ distinct binary strings of length ℓ. In Theorem 21 we show that every $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithm runs exponential time on $\mathrm{BPHP}_{2^\ell}^{2^\ell+1}$. The proof of the lower bound is rather technically involved.

2 Preliminaries

2.1 Branching programs

Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables.

A *branching program* is a directed acyclic graph with one node with in-degree 0 and out-degree 2 (source), several inner nodes with out-degree 2, and two nodes with out-degree 0 (sinks). Every node except sinks is labeled with some variable from X, one of its outgoing edges is labeled with 0 and the other one is labeled with 1. One sink is labeled with 0 and the other with 1.

Each node v in a branching program computes a Boolean function f_v. The function f_v is defined recursively for v from sinks to the source. If v is a sink labeled with $k \in \{0,1\}$ then $f_v \equiv k$. Otherwise, suppose that v is labeled with a variable x_i its outgoing edge labeled with 0 goes to a node v_0 and its outgoing edge labeled with 1 goes to a node v_1. Then we define $f_v(x_1,\dots,x_n) = f_{v_0}(x_1,\dots,x_n)$ if $x_i = 0$ and $f_v(x_1,\dots,x_n) = f_{v_1}(x_1,\dots,x_n)$ if $x_i = 1$. Since the corresponding graph is acyclic the definition is correct for each node v. We say that a branching program computes the function that corresponds to its only source.

A branching program is called *read-once* (denoted 1-BP) if each its path contains at most one occurrence of each variable.

A branching program is called an *ordered binary decision diagram* (OBDD) if variables on every path from the source to sinks appear according to some fixed order of variables.

Sometimes we write π-OBDD instead of OBDD to emphasize that variables appear according to the order of variables π.

A *nondeterministic branching program* (NBP) is a directed acyclic graph with one node with in-degree 0 and out-degree 2 (source) several inner nodes with out-degree 2 and two nodes with out-degree 0 (sinks). Each node except sinks is either *common* and is labeled with some variable from X as in the definition of branching programs or *guessing* and is not labeled. Each *common* node has two outgoing edges. One of them is labeled with 0 and the other one is labeled with 1. Each *guessing* node has two non-labeled outgoing edges. One sink is labeled with 0 and the other with 1.

As in the definition of a branching program, each node v of NBP computes a Boolean function f_v. If v is a *common* node or a sink node then f_v is defined as in the case of deterministic branching programs. If v is a *guessing* node then it has two outgoing edges. Denote heads of the edges as $\{v_0, v_1\}$. Then we define $f_v = f_{v_0} \vee f_{v_1}$.

A NBP is called *read-once* (1-NBP) if each of its path contains at most one occurrence of each variable.

Let B be a deterministic or nondeterministic branching program. Then the size of B denoted by $|B|$ is the number of nodes in the corresponding graph.

Let $f(x_1,\dots,x_n)$ be a Boolean function and let B be a deterministic or nondeterministic branching program that computes f. Let ρ be a partial substitution into X. Then $f|_\rho = f(\rho(x_1),\dots,\rho(x_n))$ is a Boolean function that depends only on variables that are not in the ρ's domain. We denote by $B|_\rho$ the result of the following (syntactic) transformations of B: we perform the transformation from bottom to top. Suppose node v is labeled with a variable x_i such that $\rho(x_i) = a \in \{0,1\}$ and its outgoing edge labeled with 0 is going to a node v_0 and its other outgoing edge is going to v_1. Then we delete v from the graph and redirect edges that are going into v to the node v_a.

It is easy to see that $B|_\rho$ computes $f|_\rho$.

Also, note that the size of $B|_\rho$ is at most the size of the B.

▶ **Lemma 1** ([4], [22]).

1. *Let A and B be two π-OBDDs for some order of variables π. And let $\odot$ be a binary operation e.g $\wedge$, $\vee$, $\oplus$, etc. There exists an algorithm that takes A and B as inputs and returns a π-OBDD that computes $A \odot B$. Moreover, the algorithm runs in time $O(|A||B|)$. Therefore its output has size $O(|A||B|)$ (see Section 4 from [4]).*
2. *Let A be a π-OBDD and let x be its arbitrary variable. Then there exists an algorithm that takes A as an input and returns a π-OBDD that computes $\exists x A$ in time $O(|A|^2)$. Therefore the size of the output is also bounded by $O(|A|^2)$. (see Section 4 from [4])*
3. *Let A be a π_1-OBDD for some variables order π_1 and let π_2 be an another arbitrary variables order. There exists an algorithm that takes A as an input and returns a minimal size π_2-OBDD B such that $A \equiv B$ i.e. it computes the same Boolean function. Moreover, this algorithm runs at most* $\mathrm{poly}(|A| + |B|)$ *steps (see Section 5 from [22]).*

▶ **Lemma 2** (Lemma 4.2 from [11]). *Let D be a 1-NBP computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and let $1 \le i \le n$. If we change every node in D labeled with the variable x_i by a guessing node and remove all labels of all its outgoing edges, then we obtain a valid 1-NBP that computes $\exists x_i f(x_1, x_2, \ldots, x_n)$.*

2.2 Proof systems

A *resolution refutation* of an unsatisfiable CNF formula φ is a sequence of clauses $C_1, C_2, \ldots, C_s$ such that (1) C_s is the empty clause (identically false), (2) for all $i \in [s]$, the clause C_i is either a clause of φ, or can be obtained by the resolution rule from two clauses with lesser numbers, where the resolution rule allows to derive $A \vee B$ from $A \vee x$ and $B \vee \neg x$. A resolution refutation is *tree-like* if every derived clause can be used as a premise of the resolution rule at most once.

Let $\varphi = \bigwedge_i C_i$ be an unsatisfiable CNF formula. An OBDD *refutation* [3, 14] of φ is a sequence of OBDDs $D_1, D_2, \ldots, D_t$ such that D_t is the constant false OBDD and for all $1 \le i \le t$ the diagram D_i either represents a clause of φ or is obtained from the previous D_j's by one of the following derivation rules.

- **Conjunction (or join)** rule allows deriving a π-OBDD for $D_1 \wedge D_2$ from π-OBDDs D_1 and D_2. We emphasize here that the conjunction rule can be only applied to OBDDs with the same order of variables.
- **Projection** ($\exists$) rule allows deriving a π-OBDD represented $\exists x A$ from a π-OBDD represented A, where x a Boolean variable.
- **Weakening** rule allows deriving a π-OBDD represented B from a π-OBDD represented A if A semantically implies B, i.e. if every satisfying assignment of A also satisfies B.
- **Reordering** rule allows deriving an OBDD represented B from an OBDD represented A if A and B semantically equivalent (note that A and B may use different variable orders).

We consider several OBDD-based proof systems that use the rules defined above. We specify the allowed rules in the brackets e.g. the proof system OBDD($\wedge$) uses only the conjunction rule meanwhile the proof system OBDD($\wedge$, weakening) uses both the conjunction and the weakening rules, etc. All proof systems use the conjunction rule.

Note that the projection rule is a special case of the weakening rule, thus, both of them are usually not included in the brackets simultaneously.

The size of a refutation is the sum of the sizes of the OBDDs from it.

We also need to define 1-NBP semantical proof systems [5] that extends OBDD proof systems.

Let $\varphi = \bigwedge_i C_i$ be an unsatisfiable CNF formula. A 1-NBP refutation of φ is a sequence of 1-NBPs $D_1, D_2, \ldots, D_t$ such that D_t is the constant false 1-NBP and for all $1 \le i \le t$ the diagram D_i either represents a clause of φ or obtained from the previous D_j's by one of the following derivation rules.

- **Conjunction (or join)** rule allows deriving a 1-NBP for $D_1 \wedge D_2$ from 1-NBPs D_1 and D_2.
- **Projection** ($\exists$) rule allows deriving a 1-NBP represented $\exists x A$ from a 1-NBP represented A where x is one of φ's boolean variables.
- **Weakening** rule allow deriving a 1-NBP represented B from a 1-NBP represented A if A semantically implies B.

As in the definition of OBDD proofs, 1-NBP proof systems can be defined with different sets of inference rules. For example NBP$(\wedge, \exists)$ uses only the conjunction and the projection rules.

We emphasize that 1-NBP proof systems are not proof systems in the sense of Cook-Reckhow [8] unless P = NP since it is NP-hard to verify the correctness of a given proof.

Let $G(V, E)$ be a graph. Let $c\colon V \to \{0, 1\}$ be a *charge function.* A *Tseitin formula* $\mathrm{T}(G, c)$ depends on the propositional variables x_e for $e \in E$. For each vertex $v \in V$ we define the parity condition of v as $P_v := \left(\sum_{e \ni v} x_e \equiv c(v) \bmod 2\right)$, where $e \ni v$ means that an edge e is incident to the vertex v. The Tseitin formula $\mathrm{T}(G, c)$ is the conjunction of parity conditions of all the vertices: $\bigwedge_{v \in V} P_v$. Tseitin formulas are represented in CNF as follows: we represent P_v in CNF in the canonical way for all $v \in V$.

▶ **Theorem 3** (Theorem 3.11 from [5]). *There exists a family of constant degree graphs G_n with n vertices such that any 1-NBP$(\wedge)$ refutation of an unsatisfiable Tseitin formula $\mathrm{T}(G_n, f)$ contains a 1-NBP of size at least $2^{\Omega(n)}$.*

2.3 OBDD algorithms for SAT

The algorithm gets as an input a CNF formula ϕ, it chooses some order π on the variables and creates both a π-ordered OBDD D (which initially is equal to the constant true function) and a set of clauses S (which initially consists of all clauses of the formula ϕ). While S is not empty the algorithm applies one of the following three operations:

- **Conjunction (or join)** delete some clause C from S and replace D by a π-OBDD that represents the conjunction $D \wedge C$;
- **Projection** ($\exists$) choose a variable x that has no occurrences in the clauses from S and replace D by a π-OBDD for the function $\exists x D$;
- **Reordering** choose a new order on variables π' and replace D by the equivalent π'-OBDD. Assign $\pi := \pi'$.

After every step of the algorithm, the following invariant is maintained: ϕ is satisfiable if and only if $\bigwedge\limits_{C \in S} C \wedge D$ is satisfiable. After the termination of the algorithm, the set S is empty; if the diagram D has a path from the source to a sink labeled by 1, then the algorithm returns "Satisfiable", otherwise it returns "Unsatisfiable".

We refer to the algorithms of this type as OBDD$(\wedge, \exists, \text{reordering})$ algorithms. Besides, we use a similar notation for algorithms that use some of the operations: we just enumerate the used operations in the brackets. For example, the OBDD$(\wedge)$ algorithms use only the conjunction operation, and the OBDD$(\wedge, \exists)$ algorithms use only the conjunction and projection operations.

Since join and projection for OBDDs may be performed in polynomial time and reordering may be performed in time polynomial in the sizes of the input and the output, the running time of an OBDD$(\wedge, \exists, \text{reordering})$ algorithm is polynomially related to the sum of the sizes of all states of the diagram D (here we ignore the time spent on choosing the next step of the algorithm).

We also define a purely theoretical notion of 1-NBP algorithms for SAT that naturally extends OBDD algorithms.

The algorithm gets as an input a CNF formula ϕ. It creates both a 1-NBP D (which initially is equal to the constant true function) and a set of clauses S (which initially consists of all clauses of the formula ϕ). While S is not empty the algorithm applies one of the following two operations:

- **Conjunction (or join)** delete some clause C from S and replace D by a 1-NBP that represents the conjunction $D \wedge C$;
- **Projection ($\exists$)** choose a variable x that has no occurrences in the clauses from S and replace D by a 1-NBP for the function $\exists x D$.

An execution track of a 1-NBP (OBDD) algorithm is the sequence of all 1-NBPs (OBDDs) constructed by the algorithm during its runtime. A *total size* of an execution track is the total size of 1-NBP (OBDDs) in it. By the running time of 1-NBP algorithms, we mean the sum of the sizes of all branching programs from its execution track. We will mainly use the lower bound of the running time of 1-NBP algorithms as a lower bound for OBDD algorithms.

▶ **Lemma 4.** *From every execution track of an* OBDD($\wedge, \exists$, *reordering*) *algorithm one can remove several diagrams to get an execution track of a correct* $1-$NBP($\wedge, \exists$) *algorithm.*

Proof. Since OBDD is a special case of $1-$NBP, applications of conjunction and join operations of OBDD($\wedge, \exists$, reordering) algorithm are legal operations for 1-NBP($\wedge, \exists$) algorithms. Notice that the reordering rule does not change the Boolean function, so we can just remove the larger of two OBDDs representing the same Boolean function. ◀

2.4 Quantified Boolean formulas

An $\exists$-CNF formula is a formula of type $\exists x_{i_1} \exists x_{i_2}, \ldots \exists x_{i_k} \phi(x_1, x_2, \ldots, x_n)$, where $\phi(x_1, x_2, \ldots, x_n)$ is a CNF formula, $\{i_1, i_2, \ldots, i_k\} \subseteq [n]$ and k is non-negative integer. The formula ϕ is called *the matrix* of the $\exists$-CNF formula.

▶ **Lemma 5.** *Let φ be an $\exists$-CNF formula. Assume that a partial assignment ρ satisfies φ. Then for every variable x, ρ satisfies $\exists x \varphi$.*

Proof. If x does not have occurrences in φ, then $\exists x \varphi$ is equivalent to φ and, therefore, it is satisfied by ρ. Otherwise $\exists x \varphi$ is semantically equivalent to $\varphi|_{x:=\rho(x)} \vee \varphi|_{x:=1-\rho(x)}$. By trivial reasons ρ satisfies $\varphi|_{x:=\rho(x)}$, hence it satisfies $\exists x \varphi$. ◀

2.5 Error-correcting codes

We will use error-correcting codes to construct hard formulas and analyze the running time of the OBDD algorithms.

By a *code* we mean a subset of binary strings with a fixed length. A code C has a distance d if for any two codewords $c_1, c_2 \in C$ the Hamming distance between c_1 and c_2 is at least d. A code $C \subseteq \{0,1\}^n$ has a relative distance δ if it has the distance δn.

A *linear code* is a set of all n-bits vectors $x = (x_1 \ldots x_n)$ from some linear subspace in $\mathbb{F}_2^n$.

A linear code can be specified by a system of linear equations. For a code of dimension k this system should consist of $m \geq n - k$ linear equations involving n variables. The set of all solutions of the system should give exactly our code, so the rank of the system must be equal to $n - k$. If we require in addition that the equations in the system are linearly independent, then the number of equations is equal to $m = n - k$. The matrix of this linear system is called a *checksum matrix* of the code.

▶ **Lemma 6** (Hamming code [13]). *There is a linear code $C \subseteq \{0,1\}^n$ of size $2^{\Omega(n/\log n)}$ with distance* 3.

For $0 < p < 1$ *the binary entropy* is $H(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$. We will use the binary entropy to estimate the size of a linear code with the given relative distance.

▶ **Lemma 7** (Gilbert-Varshamov Bound [9], [21]). *Let $0 < \varepsilon < \frac{1}{2}$. Then there exists a linear code $C \subseteq \{0,1\}^n$ of size at least $2^{(1-H(\varepsilon))n}$ with relative distance ε.*

2.6 Communication complexity

Communication complexity is one of the ways to estimate the size of OBDD representation of Boolean functions.

Let $f : A \times B \to C$ be a function. Two players Alice and Bob want to compute $f(a,b)$ for some $a \in A$ and $b \in B$. However, Alice knows only a and Bob knows only b. In order to compute the value they can use a two-sided communication channel. They agreed in advance on a protocol; at each step of the protocol, one of them sends a bit string to the other, at the end of the protocol both Alice and Bob should know $f(x)$. The cost of the protocol is the maximal number of bits they sent to each other. The communication complexity of f is equal to the minimal cost of the protocols for f (for the formal definition see [16]).

Let $f : A \times B \to C$ be a function. A set $S \subseteq A \times B$ is called a *fooling set* if exists $z \in C$ such that for all $(a,b) \in S$, $f(a,b) = z$. But for all $a_1 \neq a_2 \in A$ and $b_1 \neq b_2 \in B$ if $(a_1,b_1) \in S$ and $(a_2,b_2) \in S$, then $f(a_1,b_2) \neq z$ or $f(a_2,b_1) \neq z$.

▶ **Lemma 8** (Lemma 1.20 from [16]). *If S is a fooling set of size k for a function $f : A \times B \to C$. Then the communication complexity of f with respect to partition (A,B) is at least $\log k$.*

▶ **Lemma 9** (Lemma 12.12 from [16]). *Let $f : A \times B \to \{0,1\}$ be a Boolean function. Assume that t is the deterministic communication complexity of f with respect to partition A and B. Then for every variable order π that respects this partition (i.e. all variables from A are π-less then all variables from B or vice versa), the size of any $\pi-$OBDD computing f is at least 2^t.*

3 Lower bounds for $1-\mathrm{NBP}$ algorithms

In this section, we give a construction of a hard unsatisfiable formula for $1-\mathrm{NBP}(\wedge,\exists)$ algorithms. In Subsection 3.1 we show how to get it from a hard satisfiable formula for $1-\mathrm{NBP}(\wedge,\exists)$ algorithms and a hard formula for $1-\mathrm{NBP}(\wedge)$ proofs. In Subsection 3.2 we show that satisfiable formulas from [14] that are hard for $\mathrm{OBDD}(\wedge,\exists,\mathrm{reordering})$ algorithms are also hard for $1-\mathrm{NBP}(\wedge,\exists)$ algorithms. In Subsection 3.3 we show that $1-\mathrm{NBP}(\wedge,\exists)$ algorithms do not polynomially simulate tree-like resolution.

3.1 Hard unsatisfiable formula

Let Φ be a CNF formula and let x be a Boolean variable that does not appear in Φ. We denote by $x \vee \Phi$ the CNF formula that is obtained from Φ by adding x to each of its clauses.

Let Φ and Ψ be CNF formulas in m and n Boolean variables respectively. We define a formula $\mathcal{F}(\Phi,\Psi)$ in $n + 2mn$ variables $X = \{x_i, y_j^{(i)}, z_j^{(i)} \mid i \in [n], j \in [m]\}$ as follows:

$$\mathcal{F}(\Phi,\Psi) := \Psi(x_1,\ldots,x_n) \wedge T(X),$$

where

$$T(X) := \bigwedge_{i=1}^{n} \left(x_i \vee \Phi\left(y_1^{(i)}, y_2^{(i)}, \ldots, y_m^{(i)}\right)\right) \wedge \bigwedge_{i=1}^{n} \left(\neg x_i \vee \Phi\left(z_1^{(i)}, z_2^{(i)}, \ldots, z_m^{(i)}\right)\right).$$

Note that if Ψ is unsatisfiable then so is $\mathcal{F}(\Phi,\Psi)$.

▶ **Theorem 10.** *Let Φ be a satisfiable CNF formula in n variables and let Ψ be an unsatisfiable CNF formula in m variables. Suppose that the execution track of every $1\!-\!\mathrm{NBP}(\wedge, \exists)$ algorithm on the input $\Phi(y_1, \dots, y_m)$ contains a $1\!-\!\mathrm{NBP}$ of size at least S_1 and every 1-NBP($\wedge$) refutations of $\Psi(x_1, \dots, x_n)$ contains a $1\!-\!\mathrm{NBP}$ of size at least S_2 and $S_2 > n+1$. Then any execution track of a 1-NBP($\wedge, \exists$) algorithm on the input $\mathcal{F}(\Phi, \Psi)$ contains a $1\!-\!\mathrm{NBP}$ of size at least $\min(S_1, S_2)$.*

Proof. Consider an 1-NBP($\wedge, \exists$) algorithm. Let $B_1, B_2, \dots, B_\ell$ be its execution track on the input $\mathcal{F}(\Phi, \Psi)$. For all $i \in [\ell]$, B_i represents a $\exists$-CNF formula whose matrix is the conjunction of a subset of clauses of the input formula; we denote this $\exists$-CNF formula by F_i.

We consider two cases of what happened earlier: the matrix of F_j contains an unsatisfiable set of clauses from $\Psi(x_1, \dots, x_n)$, or F_j is quantified over some variable x_i for $i \in [n]$.

Case 1. There exists $k \in [\ell]$ such that the matrix of F_k contains an unsatisfiable set of clauses from $\Psi(x_1, \dots, x_n)$ and F_k itself is not quantified over x_i for all $i \in [n]$.

Since the formula Φ is satisfiable, there exist an assignment ρ of the variables $y_i^{(j)}, z_i^{(j)}$ for $1 \le i \le n$, $1 \le j \le m$ such that all copies of Φ in $\mathcal{F}(\Phi, \Psi)$ are satisfied by ρ.

Consider the sequence $B_1|_\rho, \dots, B_k|_\rho$. For every $i \in [k-1]$:

- if $B_{i+1} \equiv B_i \wedge C$, where C is a clause of Ψ, then $B_{i+1}|_\rho \equiv B_i|_\rho \wedge C$; here and after $\equiv$ means the semantical equivalence of Boolean functions;
- if $B_{i+1} \equiv B_i \wedge C$, where C is a clause of $T(X)$, then $B_{i+1}|_\rho \equiv B_i|_\rho$;
- if $B_{i+1} \equiv \exists z B_i$, where $z \in X \setminus \{x_1, x_2, \dots, x_n\}$, then $B_{i+1}|_\rho = B_i|_\rho$;

Let $\tilde{C}_1, \tilde{C}_2, \dots, \tilde{C}_s$ be $1\!-\!\mathrm{NBP}$ representations of all clauses of Ψ. It is easy to see that $|\tilde{C}_i| \le n+1$ for all $i \in [s]$. Then $\tilde{C}_1, \tilde{C}_2, \dots, \tilde{C}_s, B_1|_\rho, \dots, B_k|_\rho$ is the correct 1-NBP($\wedge$) refutation of Ψ. By the properties of the formula Ψ, any refutation should contain a $1\!-\!\mathrm{NBP}$ of size at least S_2. Since $S_2 > n+1$, there is $j \in [k]$ such that $|B_j|_\rho|$ is at least S_2, hence, $|B_j| \ge S_2$.

Case 2. Let p be the minimal number such that F_p is quantified with some variable x_i for $i \in [n]$ (we denote this variable by x_{i_0}). In the considered case, the matrix of F_p does not contain an unsatisfiable set of clauses from Ψ. Notice that $F_p = \exists x_{i_0} F_{p-1}$.

For $j \in [p]$, F_j is equivalent to $\Psi'_j \wedge \Theta_j$, where Ψ'_j is the satisfiable conjunction of several clauses of Ψ and Θ_j is the $\exists$-CNF formula whose matrix is the conjunction of several clauses from $T(X)$. Let $\alpha : \{x_1, x_2, \dots, x_n\} \to \{0,1\}$ be a satisfying assignment of Ψ'_{p-1}. Notice that α satisfies all Ψ'_j for $j \in [p-1]$. Let $\beta : \{t_1, t_2, \dots, t_m\} \to \{0,1\}$ be a satisfying assignment of $\Phi(t_1, t_2, \dots, t_m)$.

Let us define a partial assignment ρ to the variables X.

- $\rho(x_i) = \alpha(x_i)$, for $i \in [n]$;
- $\rho(z_i^{(j)}) = \beta(t_i)$, for $i \in [m], j \in [n] \setminus \{i_0\}$;
- $\rho(y_i^{(j)}) = \beta(t_i)$, for $i \in [m], j \in [n] \setminus \{i_0\}$.

▷ Claim 11. It is possible to delete several $1\!-\!\mathrm{NBP}$s from the sequence $B_1|_\rho, \dots, B_{p-1}|_\rho$ to get a correct execution track of 1-NBP($\wedge, \exists$) algorithm executed on $\Phi\left(y_1^{(i_0)}, \dots, y_m^{(i_0)}\right)$ if $\alpha(x_{i_0}) = 0$ and on $\Phi\left(z_1^{(i_0)}, \dots, z_m^{(i_0)}\right)$, otherwise.

Claim 11 and the property of the formula Φ imply that there exists $i \in [p-1]$ such that $|B_i|_\rho| \ge S_1$, therefore, $|B_i| \ge S_1$.

Proof of Claim 11. W.l.o.g. assume that $\alpha(x_{i_0}) = 0$. Since α satisfies Ψ'_{p-1}, α satisfies Ψ'_j for all $j \in [p-1]$. Hence ρ satisfies Ψ'_j for all $j \in [p-1]$.

Let us represent $\Theta_j = \bigwedge_{i=1}^{n} U_j^{(i)} \wedge H_j^{(i)}$, where $U_j^{(i)}$ is $\exists$-CNF formula whose matrix is the conjunction of several clauses from $\left(x_i \vee \Phi\left(y_1^{(i)}, \dots, y_m^{(i)}\right)\right)$ and $H_j^{(i)}$ $\exists$-CNF formula whose matrix is the conjunction of several clauses from $(\neg x_i \vee \Phi(z_1^{(i)}, \dots, z_m^{(i)}))$. For all $i \neq i_0$, the matrices of $U_j^{(i)}$ and $H_j^{(i)}$ are satisfied by ρ, hence by Lemma 5, $U_j^{(i)}$ and $H_j^{(i)}$ themselves are satisfied by ρ.

Since $\alpha(x_{i_0}) = 0$, ρ satisfies $H_j^{(i_0)}$. Hence, $F_j|_\rho = U_j^{(i_0)}$ and the matrix of $U_j^{(i_0)}$ is the conjunction of several clauses of $\Phi\left(y_1^{(i_0)}, \dots, y_m^{(i_0)}\right)$. Since F_p is quantified over x_{i_0}, all clauses from $\mathcal{F}(\Phi, \Psi)$ containing x_{i_0} should be in the matrix of F_{p-1}, hence the matrix of $F_{p-1}|_\rho$ is exactly $\Phi\left(y_1^{(i_0)}, \dots, y_m^{(i_0)}\right)$.

For every $j \in [p-2]$,

- if $B_{j+1} \equiv B_j \wedge C$ and $C = x_{i_0} \vee C'$, where C' is a clause of $\Phi\left(y_1^{(i_0)}, \dots, y_m^{(i_0)}\right)$, then $B_{j+1}|_\rho \equiv B_j \wedge C'$;
- if $B_{j+1} \equiv \exists y_k^{(i_0)} B_j$ and $k \in [m]$, then $B_{j+1}|_\rho \equiv \exists y_k^{(i_0)} B_j|_\rho$.
- In all other cases $B_{j+1}|_\rho \equiv B_j|_\rho$. For such j, B_{j+1} will be deleted from the sequence as required by the claim. ◁

◀

The following lemma is proved in Subsection 3.2.

▶ **Lemma 12** (cf. Corollary 5.4 from [14]). *For all large enough n there exists a satisfiable CNF formula with n Boolean variables, of size $O(n)$ such that the execution track of every $1\mathrm{-NBP}(\wedge, \exists)$ algorithm running on this formula contains a $1\mathrm{-NBP}$ of size at least $2^{\Omega(n)}$. Moreover, for a given n such a formula can be constructed by a deterministic algorithm in time $\mathrm{poly}(n)$.*

▶ **Corollary 13.** *In poly(n) time one can construct an unsatisfiable formula F_n in $\mathrm{poly}(n)$ variables such that the execution track of every $1\mathrm{-NBP}(\wedge, \exists)$ algorithm running on the formula F_n contains $1\mathrm{-NBP}$ of size at least $2^{\Omega(n)}$.*

Proof. Let Ψ_n be a Tseitin formula $T(G_n, f)$ based on the graph G_n from Theorem 3. By Theorem 3, any $1\mathrm{-NBP}(\wedge)$ refutation of Ψ_n contains a $1\mathrm{-NBP}$ of size at least $2^{\Omega(n)}$. Let Φ_n be a satisfiable formula from Lemma 12. Then we can take $F_n = \mathcal{F}(\Psi, \Phi)$ and it has the required property by Theorem 10. ◀

3.2 Hard satisfiable formulas for $1\mathrm{-NBP}(\wedge, \exists)$ algorithms

In this section, we prove Lemma 12. The proof is mainly repeating the proof from [14] for the case of $\mathrm{OBDD}(\wedge, \exists, \mathrm{reordering})$ algorithms.

We say that a code $C \subseteq \{0,1\}^n$ recovers a ρ fraction of erasures by a list of size L (or C is (ρ, L)-erasure list-decodable) if for any $w \in \{0, 1, ?\}^n$ such that the number of ? in w does not exceed ρn, there exist at most L elements in C that are consistent with w. A string $s \in \{0,1\}^n$ is consistent with w if for all i, $w_i \in \{0,1\}$ implies $s_i = w_i$.

▶ **Theorem 14** ([12, Lemma 2]). *If $\mathcal{C}$ is a code with relative distance δ, then for every $\epsilon > 0$ the code C is $((2-\epsilon)\delta, \frac{2}{\epsilon})$-erasure list-decodable.*

The following theorem states that every $1\mathrm{-NBP}$ for a characteristic function of a good enough code has at least exponential size. It extends Theorem 5.2 from [14] which claims the same lower bound on the size of OBDDs.

▶ **Theorem 15** (cf. Theorem 5.2 from [14]). *Let $C \subseteq \{0,1\}^n$ be a $(\frac{1}{2}+\epsilon, L)$-erasure list-decodable code with relative distance more than 2ϵ. Any $1-\mathrm{NBP}$ representation of the characteristic function of C (i.e. function $\chi_C\colon \{0,1\}^n \to \{0,1\} : \forall x \in \{0,1\}^n\ \chi_C(x) = 1 \Leftrightarrow x \in C$) has size at least $\frac{|C|}{L^2}$.*

Moreover, for every tuple of k different indices $i_1, \dots, i_k \in [n]$ $(0 \le k \le 2\epsilon n)$ size of any $1-\mathrm{NBP}$ representation of the Boolean function $\exists x_{i_1} \dots \exists x_{i_k}\ \chi_C(x_1, \dots, x_n)$ is at least $\frac{|C|}{L^2}$.

Proof. It is enough to prove the "moreover" part of the statement since the first part is its special case (with $k = 0$).

Notice that since any string of size $\lceil(\frac{1}{2}-\epsilon)n\rceil$ has at most L prolongation to an element of C, $|C| \le 2^{\lceil(\frac{1}{2}-\epsilon)n\rceil}L$. We may assume that $L < 2^{\lfloor(\frac{1}{2}-\epsilon)n\rfloor}$, since otherwise $\frac{|C|}{L^2} \le 2$ and the theorem is trivial.

Let D be a $1-\mathrm{NBP}$ computing $\exists x_{i_1} \dots \exists x_{i_k}\ \chi_C(x_1, \dots, x_n)$.

Consider $|C|$ codewords, for each of them there is a path in D from the source to 1-sink that is consistence with the codeword. For each such path, we mark a node v such that between the source and v there are queried exactly $\lceil\frac{n-k}{2}\rceil$ variables (the query in v is not included). We claim that such vertex always exists. Indeed, assume for the sake of contradiction that there is an accepting path p corresponding to a codeword $c \in C$ that queries t variables and $t < \lceil\frac{n-k}{2}\rceil$ variables. There are at least $2^{\lfloor\frac{n-k}{2}\rfloor}$ partial assignments from $\{x_i \mid i \in [n] \setminus \{i_1, i_2, \dots, i_k\}\} \to \{0,1\}$ such that each of them is consistent with c in the values of at least $\lceil\frac{n-k}{2}\rceil$ positions including all variables from the path p. Since all these assignments are consistent with p, they are accepted by D, hence each of them may be continued to a codeword. Hence there are at least $2^{\lfloor\frac{n-k}{2}\rfloor}$ codewords that agree with c in at least $\lceil\frac{n-k}{2}\rceil$ positions. Hence, $2^{\lfloor(\frac{1}{2}-\epsilon)n\rfloor} \le 2^{\lfloor\frac{n-k}{2}\rfloor} \le L$. This contradicts our assumptions on the size of L.

Let us estimate from the above the number of times that the same node can be marked. We claim that every marked node v can be marked at most L^2 times. Assume for the sake of contradiction that there is a node v that is marked at least $L^2 + 1$ times. Let $S \subseteq C$ be a set of codewords such that the node v was marked on the paths corresponding to them. Consider a codeword $s \in S$ and the path which is consistence with s, let between the source and v (not including the query in v) the set of queried variables be equal to $\{x_i \mid i \in I\}$. Let $J = ([n] \setminus \{i_1, i_2, \dots, i_k\}) \setminus I$. Since the relative distance of C is more than 2ϵ, no two codewords coincide on $([n] \setminus \{i_1, i_2, \dots, i_k\})$. Hence, by the pigeonhole principle, the set S contains at least $L + 1$ elements with different projections on the set I, or at least $L + 1$ elements with different projections on J. Consider these cases separately.

Assume that S contains $L + 1$ elements with different projections on I: $s, s_1, \dots, s_L$. Consider partial assignments $\tau_1, \tau_2, \dots, \tau_L : \{x_i \mid i \in I \cup J\} \to \{0,1\}$, where for all $j \in [L]$ and for all $i \in I$, $\tau_j(x_i)$ equals the ith bit of s_j and for all $i \in J$ and $\tau_j(x_i)$ equals the ith bit of s. Since s is a codeword, it is accepted by D; for all $i \in [L]$, τ_i is also accepted by D by the following accepting path: from the source to v we follow the path corresponding s_i and from v to 1-sink we follow the path corresponding s. Hence for every $j \in [\ell]$, τ_j can be extended to a codeword t_j. By the construction $s, \tau_1, \dots, \tau_L$ coincide in the set of positions J, i.e., in $\frac{n-k}{2} \ge (\frac{1}{2}-\epsilon)n$ bits, hence there exists $L + 1$ different codewords that coincide in $(\frac{1}{2}-\epsilon)n$ positions, this contradicts the property of the code.

In the second case, S contains $L+1$ elements with different projections on J: $s, s_1, \dots, s_\ell$. This case can be handled analogously to the previous one. The only difference is the definition of τ_j for $j \in [\ell]$. For $i \in I$, $\tau(x_i)$ equals the ith bit of s and for $j \in I$, $\tau(x_i)$ equals the ith bit of s_j.

So we get that there are at least $\frac{|C|}{L^2}$ distinct marked nodes in D. ◀

Using Theorem 15 we extend Theorem 5.3 from [14] from $\mathrm{OBDD}(\wedge, \exists, \text{reordering})$ algorithms to $1\text{−}\mathrm{NBP}(\wedge, \exists)$ algorithms.

▶ **Theorem 16** (cf. Theorem 5.3 from [14]). *Let $C \subseteq \{0,1\}^n$ be a linear code with the relative distance $\frac{1}{3}$ such that the checksum matrix H of the code C has the following properties:*

- *H is a binary matrix of size $\alpha n \times n$, where $\alpha \in (0,1)$ is a constant;*
- *every row of H contains at most $t(n)$ ones, where t is some function;*
- *in every $\frac{1}{6}n$ columns of H we can find ones in at least $(\alpha - \delta)n$ different rows of the matrix, where $\delta \in \left(0, \min\{\alpha, \frac{1-\alpha}{2}\}\right)$ is a constant.*

Denote by F_n the canonical CNF representation of the system of linear equations $H(x) = 0$; F_n is in $t(n)$-CNF, hence the size of F_n is at most $\alpha n 2^{t(n)-1} t(n)$. Then the execution track of every $1\text{−}\mathrm{NBP}(\wedge, \exists)$ algorithm running on F_n contains a $1\text{−}\mathrm{NBP}$ of size at least $2^{\Omega(n)}$.

Proof. The proof resembles the proof of Theorem 5.3 from [14]. The only source of OBDD size lower bound in that proof is the usage of Theorem 5.2 from [14]; we will use Theorem 15 instead to get a lower bound on the size of $1\text{−}\mathrm{NBP}$. ◀

Lemma 12 follows from Theorem 16 applied to the codes constructed in [14].

3.3 Comparison with tree-like Resolution

▶ **Definition 17.** *Let $\mathcal{P}$ be a proof system used to derive refutations of CNF formulas. A set E of extension axioms for a set of propositional variables $\vec{x}$ is a set of clauses expressing $z_i := \psi_i(\vec{x}, z_1, \ldots, z_{i-1})$, where each ψ_i is a conjunction of literals and $z_1, \ldots, z_\ell$ are new variables. Let $\varphi(\vec{x})$ be a CNF formula. Then, an* extension-$\mathcal{P}$ *refutation of $\varphi(\vec{x})$ is by definition a refutation of $\varphi \wedge E$ where E is a set of extension axioms for $\vec{x}$.*

For example, a definition by extension of the form $z := (y_1 \wedge y_2)$ is represented by the three clauses $\neg z \vee y_1$, $\neg z \vee y_2$, and $\neg y_1 \vee \neg y_2 \vee z$.

▶ **Lemma 18.** *Let φ be a CNF formula and let E be a set of extension rules represented in CNF. Then for any execution track of $1\text{−}\mathrm{NBP}(\wedge, \exists)$ algorithm on the input $\varphi \wedge E$ exists a correct execution track of $1\text{−}\mathrm{NBP}(\wedge, \exists)$ algorithm on the input φ of not greater total size.*

Proof. It is sufficient to show it for the case when E contains just one extension rule. We assume that E contains the only rule $z := \psi(y_1, \ldots, y_k)$, where y_i are variables of φ. Let $D_1, D_2, \ldots, D_s$ be an execution track of an $1\text{−}\mathrm{NBP}(\wedge, \exists)$ algorithm on the formula $\varphi \wedge E$.

Every D_i represent a $\exists$-CNF formula with a matrix $(\Phi \wedge \Psi)$, where Φ is a conjunction of clauses of φ, and Ψ is a conjunction of several clauses from E.

Notice that since E is a CNF representation of $z := \psi(y_1, \ldots, y_k)$, then $\exists z \Psi$ is identically true. Since z has no occurrences in φ, $\exists z D_i$ is equivalent to $\exists x_{i_1} \ldots \exists x_{i_\ell} \Phi$.

By Lemma 2 there exists a $1\text{−}\mathrm{NBP}$ representing $\exists D_i$ of size at most the size of D_i. It is easy to see that $\exists z D_1, \exists z D_2, \ldots, \exists z D_s$ may be considered (possibly after deletion of several extra $1\text{−}\mathrm{NBP}$s) as a correct execution track of a $1\text{−}\mathrm{NBP}(\wedge, \exists)$ algorithm on the formula φ. ◀

▶ **Lemma 19** (Lemma 4.6 from [5]). *Let G_n be an undirected graph with n vertices, with all vertices of degree at most d. Let f_n be a labeling function for G_n such that the Tseitin formula $\mathrm{T}(G_n, f_n)$ is unsatisfiable. Then there is a set E of extension axioms for $\mathrm{T}(G_n, f_n)$ of size $\mathrm{poly}(n)$ such that there is a tree-like resolution refutation of $\mathrm{T}(G_n, f_n) \wedge E$ of size $\mathrm{poly}(n)$.*

▶ **Theorem 20.** *There is a family of CNF formulas φ_n of size* $\mathrm{poly}(n)$ *such that φ_n has poly(n) tree-like resolution refutation but any execution track of* $1\text{−NBP}(\wedge, \exists)$ *algorithm on φ_n contains a* 1−NBP *of size* $2^{\Omega(n)}$.

Proof. Consider a hard formula F_n from Corollary 13. F_n has a form $\mathrm{T}(G_n, f_n) \wedge \psi$, where ψ is a satisfiable formula. By Lemma 19, there exists a set E of extension axioms for $\mathrm{T}(G_n, f_n)$ of size $\mathrm{poly}(n)$ such that there is a tree-like resolution refutation of $\mathrm{T}(G_n, f_n) \wedge E$ of size $\mathrm{poly}(n)$. Notice that $\mathrm{T}(G_n, f_n) \wedge E \wedge \psi$ also has a tree-like resolution refutation of size $\mathrm{poly}(n)$. By Corollary 13 and Lemma 18, the execution track of any $1\text{−NBP}(\wedge, \exists)$ algorithm on the formula $\mathrm{T}(G_n, f_n) \wedge E \wedge \psi$ contains a 1−NBP of size $2^{\Omega(n)}$. ◀

4 Binary pigeonhole principle is hard for $\mathrm{OBDD}(\wedge, \exists, \text{reordering})$ algorithms

The Binary Pigeonhole Principle represents in CNF that it is impossible to have $2^l + 1$ distinct binary strings of length l. Let $X = \{X_{i,j} \mid i \in [2^l + 1], j \in [l]\}$ be a set of propositional variables. For every $i \in [2^l + 1]$ we denote the vector of variables $(X_{i,1}, \ldots, X_{i,l})$ as X_i. For all distinct i and j from $[2^l + 1]$ and for all binary strings $a \in \{0, 1\}^l$ we define a clause $C^a_{i,j}$ that encodes that at least one of the strings X_i or X_j differs from a as follows $C^a_{i,j} = \bigvee_{m=1}^{l} (X_{i,m} \neq a_m \vee X_{j,m} \neq a_m)$, where for a propositional variable x, $x \neq 0$ denotes x and $x \neq 1$ denotes $\neg x$. Let us denote $[X_i \neq X_j] := \bigwedge_{a \in \{0,1\}^l} C^a_{i,j}$ the CNF formula that encodes that $X_i \neq X_j$. We finally define $\mathrm{BPHP}^{2^l+1}_{2^l} := \bigwedge_{i,j \in [2^l+1]: i \neq j} [X_i \neq X_j]$.

It is convenient to consider X as a $(2^l + 1) \times l$ matrix of propositional variables. In this paper, we refer to X as *the variables matrix* or simply *the matrix* and to X_i for $i \in [2^l + 1]$ as the ith row of the matrix.

The main result of this section is the following.

▶ **Theorem 21.** *Let $n = 2^l$ and $0 < \varepsilon < 1$ be a solution of the equation $\varepsilon = 1 - H(\varepsilon)$ ($\varepsilon \approx 0.227092$). Then the execution track of any* $\mathrm{OBDD}(\wedge, \exists, \textit{reordering})$ *algorithm on the input* BPHP^{n+1}_n *contains an* OBDD *of size at least* $2^{\Omega(n^\varepsilon / \log n)}$.

Proof. Consider an $\mathrm{OBDD}(\wedge, \exists, \text{reordering})$ algorithm $\mathcal{A}$ and its execution on the input BPHP^{n+1}_n.

We consider several cases:

Case 1. Suppose that during its running the algorithm $\mathcal{A}$ applied projections to variables that all lie in at most $n/2$ different rows of the variables matrix X. Consider the state of the diagram D before the last application of the conjunction operation. In this moment D represents an $\exists$-CNF formula whose matrix is the conjunction of all but one clauses of BPHP^{n+1}_n. The following lemma implies that in this case $|D| \geq 2^{\Omega(n/\log n)}$.

▶ **Lemma 22.** *Let $\mathcal{C}$ be the strict subset of the set of clauses of* BPHP^{n+1}_n. *Let $Z = \{z_1, \ldots, z_b\} \subseteq X$ be a set of* BPHP^{n+1}_n *variables. Assume that there is a set $Y \subseteq [n+1]$ such that*

1. *for all $i \in Y$, X_i does not contain variables from Z;*
2. *$|Y| \geq n/\log n$;*
3. *for all $i, j \in Y$, $\mathcal{C}$ contains all clauses representing $[X_i \neq X_j]$.*

Then any OBDD *representing $\varphi := \exists z_1, \ldots, z_b \bigwedge_{C \in \mathcal{C}} C$ has size at least $2^{\Omega(n/\log n)}$.*

We prove Lemma 22 in Subsection 4.1.

Case 2. The algorithm $\mathcal{A}$ applied projection operation over variables from more than $n/2$ rows of the variables matrix X. Since $\mathcal{A}$ applies projection operations one by one, one of the following events happens before the other:

1. There is a row of the variable's matrix X such that there are $\lfloor \varepsilon l \rfloor - 3$ variables in this row for which $\mathcal{A}$ applied projection operations.
2. $\mathcal{A}$ applies projections over variables from $\lfloor n^\varepsilon / 100 \log n \rfloor$ rows of the variables matrix X.

Case 2.1. Assume that the first event happened before the second. Consider the state of the diagram D right after $\mathcal{A}$ first time has applied projection over $\lfloor \varepsilon l \rfloor - 3$ variables from some row. Since $\varepsilon < 1/4$, the next lemma implies that $|D| \geq 2^{\Omega(n^\varepsilon / \log n)}$.

▶ **Lemma 23.** *Let $\mathcal{C}$ be a proper subset of clauses from BPHP_n^{n+1}. Let $Z = \{z_1, \dots, z_b\} \subseteq X$ be a set of BPHP_n^{n+1} variables. Suppose that for all $z \in Z$ all occurrences of variables of Z in BPHP_n^{n+1} are in clauses from $\mathcal{C}$. Suppose also that:*

1. *There exists a row of X containing exactly k variables from Z.*
2. *Variables from Z occur in at most $n/2$ rows of X.*

Then every OBDD that computes $\varphi := \exists z_1, \dots, z_b \bigwedge_{C \in \mathcal{C}} C$ has size at least $\min\left\{2^{\Omega(2^k/k)}, 2^{\Omega\left(2^{l-3k}/l^2\right)}\right\}$.

We prove Lemma 23 in Subsection 4.2.

Case 2.2. Now assume that the second event happened before the first. Consider the state of the diagram D right after $\mathcal{A}$ first time has applied projection over variables from $\lfloor n^\varepsilon / 100 \log n \rfloor$ different rows.

▶ **Lemma 24.** *Let $\mathcal{C}$ be a strict subset of the set of clauses of BPHP_n^{n+1}. Let $Z = \{z_1, \dots, z_b\} \subseteq X$ be a set of BPHP_n^{n+1} variables. Suppose that for all $z \in Z$ all clauses BPHP_n^{n+1} having variables from Z are in $\mathcal{C}$. Let $d \in [l-1]$ and $T \in [2^l]$ be such that*

1. *variables from Z occur in exactly T rows of the variables matrix X;*
2. *each row of X contains at most d variables from Z;*
3. *there exists a code $ECC \subseteq \{0,1\}^l$ with distance $d+3$ and size $|ECC| = T+1$;*
4. *$1 + (l+1) \cdot T \cdot 2^{d+1} < n/2$.*

Then every OBDD that computes $\varphi := \exists z_1, \dots, z_b \bigwedge_{C \in \mathcal{C}} C$ has size at least 2^T.

The proof of Lemma 24 is given in Subsection 4.3.

By Lemma 7, there exists a code $ECC \subseteq \{0,1\}^l$ with distance $\varepsilon\ell$ and size $2^{(1-H(\varepsilon))\ell} = n^{1-H(\varepsilon)} = n^\varepsilon$, hence there exists code with the same distance of size $T := \lfloor n^\varepsilon / (100 \log n) \rfloor + 1$. It is straightforward that $1 + (l+1) \cdot T \cdot 2^{d+1} < n/2$. Hence, Lemma 24 implies that $|D| \geq 2^{\Omega(n^\varepsilon / \log n)}$. ◀

4.1 Proof of Lemma 22

▶ **Lemma 25.** *Let $\mathcal{C}$ be a subset of clauses from BPHP_n^{n+1}, where $n = 2^l$. Assume that $i_0, j_0 \in [n+1]$ and $a_0 \in \{0,1\}^\ell$ are such that the clause $(X_{i_0} \neq a_0 \vee X_{j_0} \neq a_0)$ does not belong to $\mathcal{C}$. Let $Z = \{z_1, \dots, z_b\} \subseteq X$ be a set of variables of BPHP_n^{n+1}. Let the set of variables X be partitioned into two parts $X^{(1)}$ and $X^{(2)}$. There are $2t$ rows of the variables matrix: $s_1^1, s_1^2, s_2^1, s_2^2 \dots, s_t^1, s_t^2 \in [n+1] \setminus \{i_0, j_0\}$ such that*

1. *for all $i \in [t]$, the rows s_i^1 and s_i^2 does not contain variables from Z.*
2. *for all $i \in [t]$, the row s_i^1 contains at least one variable from the first part $X^{(1)}$ and the row s_i^2 contains only variables from the second part $X^{(2)}$;*

3. *There are t binary strings $a_1, a_2, \dots, a_t \in \{0,1\}^\ell \setminus \{a_0\}$ such that the Hamming distance between a_i and a_j is at least 3 and for all $i \in \{0, 1, \dots, t\}$ the clause $(X_{s_i^1} \neq a_i \vee X_{s_i^2} \neq a_i)$ belongs $\mathcal{C}$.*

Then the communication complexity of computing $\varphi := \exists z_1, \dots, z_b \bigwedge_{C \in \mathcal{C}} C$ with respect to the partition $(X^{(1)} \setminus Z, X^{(2)} \setminus Z)$ is at least t.

Proof. Let us construct a fooling set of size 2^t. The elements of our fooling set are indexed by a binary string $r \in \{0,1\}^t$. Let us construct an assignment $\sigma_r : X \to \{0,1\}$ corresponding to r.

Let for $k \in [\ell]$, e_k denote the element of $\{0,1\}^l$ with $\ell - 1$ zeros and the only one on the kth place.

Let for $i \in [t]$, $k(i) := \min\{j : X_{s_i^1, j}\} \in X^{(1)}$.

- $\sigma_r(X_{s_i^1}) := a_i + (e_{k(i)} \cdot r_i)$ (here we add vectors in $\mathbb{F}_2^l$);
- $\sigma_r(X_{s_i^2}) := a_i + (e_{k(i)} \cdot (1 + r_i))$;
- $\sigma_r(X_{i_0}) := a_0$ and $\sigma_r(X_{j_0}) := a_0$.
- There are $n - 2t - 1$ other rows of the variables matrix X, let us choose some bijection between them and $\{0,1\}^\ell \setminus \{a_1, a_1 + e_{k(1)}, a_2, a_2 + e_{k(1)}, \dots, a_t, a_t + e_{k(1)}, a_0\}$ (the two sets have the same size since $\{a_0, a_1, \dots, a_t\}$ is a code with distance at least 3 and hence all binary strings in $\{a_1, a_1 + e_{k(1)}, a_2, a_2 + e_{k(1)}, \dots, a_t, a_t + e_{k(1)}, a_0\}$ are distinct) and σ_r substitutes values of the variables from these rows according to this bijection.

We claim that $\{\sigma_r \text{ restricted to } X \setminus Z \mid r \in \{0,1\}^t\}$ is a fooling set for φ. Indeed, since $\{a_0, a_1, \dots, a_t\}$ is a code with distance at least 3, for all $r \in \{0,1\}^t$, σ_r satisfies all clauses of BPHP_n^{n+1} but $(X_{i_0} \neq a_0 \vee X_{j_0} \neq a_0)$. Hence, σ_r satisfies $\bigwedge_{C \in \mathcal{C}} C$ and, then by Lemma 5, σ_r satisfies φ.

Let p and q be different strings from $\{0,1\}^t$. W.l.o.g. assume that there exists $m \in [t]$ such that $p_m = 0$ and $q_m = 1$. Let σ' be an assignment that coincides with σ_p on $X^{(1)}$ and coincides with σ_q on $X^{(2)}$. Notice that $\sigma_p(X_{s_m^1})$ and $\sigma_q(X_{s_m^1})$ differs only on $k(m)$th bit corresponding to the variable from $X^{(1)}$, hence $\sigma'(X_{s_m^1}) = \sigma_p(X_{s_m^1}) = a_m$. Analogously, $\sigma'(X_{s_m^2}) = \sigma_q(X_{s_m^1}) = a_m$. Hence σ' falsifies clause $(X_{s_m^1} \neq a_m \vee X_{s_m^2} \neq a_m)$. This clause is in $\mathcal{C}$ and all the variables of this clause are not in Z. Hence σ' falsifies φ.

So we have verified that $\{\sigma_r \text{ restricted to } X \setminus Z \mid r \in \{0,1\}^t\}$ is a fooling set of size 2^t. Hence, by Lemma 8, the communication complexity of φ is at least t. ◀

▶ **Lemma 22.** *Let $\mathcal{C}$ be the strict subset of the set of clauses of BPHP_n^{n+1}. Let $Z = \{z_1, \dots, z_b\} \subseteq X$ be a set of BPHP_n^{n+1} variables. Assume that there is a set $Y \subseteq [n+1]$ such that*

1. *for all $i \in Y$, X_i does not contain variables from Z;*
2. *$|Y| \geq n / \log n$;*
3. *for all $i, j \in Y$, $\mathcal{C}$ contains all clauses representing $[X_i \neq X_j]$.*

Then any OBDD representing $\varphi := \exists z_1, \dots, z_b \bigwedge_{C \in \mathcal{C}} C$ has size at least $2^{\Omega(n/\log n)}$.

Proof of Lemma 22. Let D be an OBDD computing φ. Let π be an extension of the variables order of D to X. Consider a partition of X on two parts with respect to π such that there are exactly $\lceil |Y|/2 \rceil$ rows from Y that have at least one variable from the first part. Let $i_0, j_0 \in [n+1]$ and $a_0 \in \{0,1\}^\ell$ be such that the clause $(X_{i_0} \neq a_0 \vee X_{j_0} \neq a_0)$ does not belong to $\mathcal{C}$. Let $H \subseteq \{0,1\}^l$ be a Hamming code (see Lemma 6), $|H| \geq \Omega(n/\log n)$. $H + a_0$ is also a code of distance at least 3 and $a_0 \in H + a_0$. So we can choose distinct strings $a_1, a_2, \dots, a_t \in H + a_0 \setminus \{a_0\}$ such that $t = \Omega(n/\log n)$. There are at least $\lceil |Y|/2 \rceil - 2$ rows in $Y \setminus \{i_0, j_0\}$ that have at least one variable from the first part. And also there are at least

$\lceil |Y|/2 \rceil - 2$ rows in $Y \setminus \{i_0, j_0\}$ with all variables lying in the second part. Since $\mathcal{C}$ contains all clauses representing $[X_i \neq X_j]$ for $i \neq j \in Y$, we can apply Lemma 25 and get that the communication complexity of φ with respect to the partition is at least t. Since the partition respects the variable order of D, by Lemma 9, $|D| \geq 2^t = 2^{\Omega(n/\log n)}$. ◀

▶ **Corollary 26.** *Any* OBDD *representing* $\bigwedge_{1<i<j\leq n+1}[X_i \neq X_j]$ *has size at least* $2^{\Omega(n/\log n)}$.

4.2 Proof of Lemma 23

▶ **Lemma 27.** *Let* $F(X_2, \dots, X_{n+1}) = \exists X_1 \bigwedge_{i=2}^{n+1}[X_1 \neq X_i]$*. Then the size of any* OBDD *for* F *is at least* $2^{\Omega(n/\log n)}$.

Proof. For all $s_2, \dots, s_{n+1} \in \{0,1\}^l$ the equality $F(s_2, \dots, s_{n+1}) = 1$ holds if and only if there exists a binary string $s_1 \in \{0,1\}^l$ that differs from $s_2, \dots, s_{n+1} \in \{0,1\}^l$. Such string exists if and only if there exist two equal strings among $s_2, \dots, s_{n+1}$. Hence $F(X_2, \dots, X_{n+1})$ is semantically equivalent to $\neg \bigwedge_{1<i<j\leq n+1}[X_i \neq X_j]$. Hence, by Corollary 26, size of any OBDD representing F is at least $2^{\Omega(n/\log n)}$. ◀

▶ **Lemma 23.** *Let* $\mathcal{C}$ *be a proper subset of clauses from* BPHP_n^{n+1}*. Let* $Z = \{z_1, \dots, z_b\} \subseteq X$ *be a set of* BPHP_n^{n+1} *variables. Suppose that for all* $z \in Z$ *all occurrences of variables of* Z *in* BPHP_n^{n+1} *are in clauses from* $\mathcal{C}$*. Suppose also that:*

1. *There exists a row of* X *containing exactly* k *variables from* Z*.*
2. *Variables from* Z *occur in at most* $n/2$ *rows of* X*.*

Then every OBDD *that computes* $\varphi := \exists z_1, \dots, z_b \bigwedge_{C\in\mathcal{C}} C$ *has size at least* $\min\left\{2^{\Omega(2^k/k)}, 2^{\Omega\left(2^{l-3k}/l^2\right)}\right\}$.

Proof of Lemma 23. Let Y be the set of rows of X that do not contain variables from Z. By the condition of the lemma, $|Y| \geq n/2 + 1$. We consider two cases.

Case 1. Assume that there is $T \subseteq Y$ such that $|T| = 2^k$ and the set $A_T := \{a \in \{0,1\}^l \mid \exists i, j \in T$ such that the clause $(X_i \neq a) \vee (X_j \neq a)$ belongs to $\mathcal{C}\}$ has size less than 2^{l-k}.

In this case, we show that there exists a partial substitution ρ such that $\varphi|_\rho$ is exactly the formula from Lemma 27 applied to variables of $\mathrm{BPHP}_{2^k}^{2^k+1}$. Hence, size of any OBDD computing $\varphi|_\rho$ is at least $2^{\Omega(2^k/k)}$ and, thus, any OBDD computing φ has size at least $2^{\Omega(2^k/k)}$.

Without loss of generality, assume that the row containing exactly k variables from Z is the first row of the variables matrix and $X_{1,1}, \dots, X_{1,k} \in Z$.

Since $|A_T| < 2^{l-k}$, there exists $b \in \{0,1\}^{l-k}$ such that there are no elements in A_T with the suffix b. Let us define ρ as follows:

- For all $i \in T \cup \{1\}$, ρ assigns b to $(X_{i,k+1}, X_{i,k+2}, \dots, X_{i,l})$.
- For $i \in [n+1] \setminus (T \cup \{1\})$, ρ assigns to X_i different elements of $\{a \in \{0,1\}^l \mid b$ is not the suffix of $a\}$.

We split $\mathcal{C}$ into three parts $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$, where

- $\mathcal{C}_1$ consists of all clauses of $\mathcal{C}$ with all variables from the set of rows T;
- $\mathcal{C}_2$ consists of all clauses of $\mathcal{C}$ containing variables from the first row and from some row from T;
- $\mathcal{C}_3$ consists of all clauses of $\mathcal{C}$ containing variables from the set of rows $[n+1] \setminus (T \cup \{1\})$.

By the properties of φ, φ is equivalent to

$$\bigwedge_{C\in\mathcal{C}_1} C \wedge \exists X_{1,1}\ldots\exists X_{1,k}\left(\bigwedge_{C\in\mathcal{C}_2} C \wedge \underset{z\in Z\setminus\{X_{1,1},\ldots,X_{1,k}\}}{\exists} z \bigwedge_{C\in\mathcal{C}_3} C\right).$$

Since the support of ρ does not contain variables from $\{X_{1,1},\ldots,X_{1,k}\}$, $\varphi|_\rho$ is equivalent to

$$\bigwedge_{C\in\mathcal{C}_1} C|_\rho \wedge \exists X_{1,1}\ldots\exists X_{1,k}\left(\bigwedge_{C\in\mathcal{C}_2} C|_\rho \wedge \left(\underset{z\in Z\setminus\{X_{1,1},\ldots,X_{1,k}\}}{\exists} z \bigwedge_{C\in\mathcal{C}_3} C\right)\Bigg|_\rho\right).$$

Since all $a \in A_T$ do not have suffix b, for all $C \in \mathcal{C}_1$, $C|_\rho = 1$.

Consider a clause $C := (X_i \neq a) \vee (X_j \neq a) \in \mathcal{C}_3$. If $i, j \in [n+1] \setminus (T \cup \{1\})$, then ρ substitute to X_i and X_j different values, hence $C|_\rho = 1$. If $i \in (T \cup \{1\})$ and $j \in [n+1] \setminus (T \cup \{1\})$, then ρ substitutes b to $(X_{i,k+1},\ldots X_{i,l})$ and something different from b to $(X_{j,k+1},\ldots X_{j,l})$, hence $C|_\rho = 1$. Thus, ρ satisfies $\bigwedge_{C\in\mathcal{C}_3} C$, hence by Lemma 5, ρ satisfies $\underset{z\in Z\setminus\{X_{1,1},\ldots,X_{1,k}\}}{\exists} z \bigwedge_{C\in\mathcal{C}_3} C$.

So we get that $\varphi|_\rho$ is equivalent to $\exists X_{1,1}\ldots\exists X_{1,k} \bigwedge_{C\in\mathcal{C}_2} C|_\rho$.

Since $\mathcal{C}$ contains all clauses of BPHP_n^{n+1} with variables from Z, $\mathcal{C}_2 = \bigwedge_{i\in T, a\in\{0,1\}^l}(X_1 \neq a \vee X_i \neq a)$. If b is not a suffix of a, then ρ satisfies $(X_1 \neq a \vee X_i \neq a)$. Hence, $\bigwedge_{C\in\mathcal{C}_2} C|_\rho$ is equivalent to $\bigwedge_{i\in T, a\in\{0,1\}^k}((X_{1,1},\ldots,X_{1,k}) \neq a \vee (X_{i,1},\ldots,X_{i,k}) \neq a)$. So $\varphi|_\rho$ satisfies the conditions of Lemma 27.

Case 2. Assume that for all $T \subseteq Y$ such that $|T| = 2^k$, the set $A_T := \{a \in \{0,1\}^l \mid \exists i, j \in T$ such that the clause $((X_i \neq a) \vee (X_j \neq a))$ belongs to $\mathcal{C}\}$ has size at least 2^{l-k}. Hence, for every $T \subseteq Y$ such that $|T| = 2^k$ there are $i, j \in T$ such that the set $A_{i,j} := \{a \in \{0,1\}^l \mid$ the clause $((X_i \neq a) \vee (X_j \neq a))$ belongs to $\mathcal{C}\}$ has size at least 2^{l-3k}.

In this case, we will obtain a lower bound by Lemma 25.

Let $i_0, j_0 \in [n+1]$ and $a_0 \in \{0,1\}^\ell$ be such that the clause $(X_{i_0} \neq a_0 \vee X_{j_0} \neq a_0)$ does not belong to $\mathcal{C}$.

Let D be an OBDD computing φ. Let π be an extension of the variables order of D to X. Let $W = Y \setminus \{i_0, j_0\}$; $|W| \geq n/2 - 1$. Consider the following order on the set of rows $[n+1]$: we say that ith row is less than jth row if the π-minimal variable of X_i is π-less than the π-minimal variable of X_j. Let us order W according to this order: $W = \{w_1, w_2, \ldots, w_{|W|}\}$. Let $d := \left\lfloor \frac{|W|}{2^k} \right\rfloor$. For every $i \in [d]$, consider the set $L_i = \{w_i, w_{i+d}, w_{i+2d}, \ldots, w_{i+(2^k-1)d}\}$. Since for all $i \in [d]$, $|L_i| = 2^k$ and $L_i \subseteq Y$, hence there are $e_i, f_i \in [2^k]$ such that $e_i < f_i$ and $|A_{w_{i+(e_i-1)d}, w_{i+(f_i-1)d}}| \geq 2^{l-3k}$. By the pigeonhole principle, there is $f \in [2^k]$ such that $|\{i \in [d] \mid f_i = f\}| \geq \frac{d}{2^k}$. Let us denote $I := \{i \in [d] \mid f_i = f\}$; $|I| \geq \frac{d}{2^k} \geq 2^{l-2k-1} - 1$.

Let us split the set X into two parts according to π: the first part consists of all variables that are π-less than the π-minimal variable of the row $w_{1+(f-1)d}$ and the second part consists of all other elements. By the construction for all $i \in I$, the row $s_i^1 := w_{i+(e_i-1)d}$ contains at least one variable from the first part (π-minimal variable of this row) and all variables from the row $s_i^2 := w_{i+(f-1)d}$ are in the second part.

To apply Lemma 25, we need to show that there exist $a_i \in A_{s_i^1, s_i^2}$ such that a_0 and a_i for $i \in I$ are on the pairwise Hamming distance at least 3. We will choose them one by one; assume that we have already chosen $a_0, a_1 \in A_{s_{i_1}^1, s_{i_1}^2}, \ldots, a_q \in A_{s_{i_q}^1, s_{i_q}^2}$ such that Hamming distance between each pair is at least 3. If $(1 + l + l(l-1)/2)q < 2^{l-3k}$, then we can choose $a_{q+1} \in A_{s_{i_{q+1}}^1, s_{i_{q+1}}^2}$. So we can choose t elements if $t = \left\lfloor \frac{2^{l-3k}}{1+l+l(l-1)/2} \right\rfloor - 1$.

Hence, by Lemma 25, the communication complexity of φ with respect to the descried partition of X is at least $\Omega\left(2^{l-3k}/l^2\right)$. Then by Lemma 9, the size of D is at least $2^{\Omega\left(2^{l-3k}/l^2\right)}$. ◀

4.3 Proof of Lemma 24

▶ **Lemma 24.** *Let $\mathcal{C}$ be a strict subset of the set of clauses of* BPHP_n^{n+1}. *Let $Z = \{z_1, \dots, z_b\} \subseteq X$ be a set of* BPHP_n^{n+1} *variables. Suppose that for all $z \in Z$ all clauses* BPHP_n^{n+1} *having variables from Z are in $\mathcal{C}$. Let $d \in [l-1]$ and $T \in [2^l]$ be such that*
1. *variables from Z occur in exactly T rows of the variables matrix X;*
2. *each row of X contains at most d variables from Z;*
3. *there exists a code $ECC \subseteq \{0,1\}^l$ with distance $d+3$ and size $|ECC| = T+1$;*
4. $1 + (l+1) \cdot T \cdot 2^{d+1} < n/2$.

Then every OBDD *that computes $\varphi := \exists z_1, \dots, z_b \bigwedge_{C \in \mathcal{C}} C$ has size at least 2^T.*

Proof of lemma 24. There exist $i_0, j_0 \in [n+1]$ and a binary string $a_0 \in \{0,1\}^l$ such that $\mathcal{C}$ does not contain the clause $(X_{i_0} \neq a_0 \vee X_{j_0} \neq a_0)$.

Let $ECC \subseteq \{0,1\}^l$ be a code with distance at least $d+3$ such that $|ECC| = T+1$. W.l.o.g. assume that $a_0 \in ECC$. Let us denote $A = ECC \setminus \{a_0\}$ and $A = \{a_1, \dots, a_T\}$.

Consider an ordered binary decision diagram D computing φ. Let π be the variables order using in D extended to all variables X. We say that a variable $x \in X$ is π-first if $x \notin Z$ and it has the minimal π-number among all such variables in its row.

As in Lemma 23 we define an order on the set of rows of the variables matrix X (or equivalently on the set $[n+1]$): ith row is less than the jth if the π-first variable of X_i is π-less the π-first variable of X_j. Let us order the elements of $[n+1] \setminus \{i_0, j_0\}$ according to this order: $\{s_1, \dots, s_{n-1}\}$.

Now we split the variables of X into two parts such that all variables in the first part precede all variables in the second part according to the order π. The first part consists of all variables that are π-less-or-equal to the π-first variable of the row number $s_{\lfloor (n-1)/2 \rfloor}$. The second part consists of the other variables.

To prove the lower bound we will define 2^T partial substitutions ρ_α defined on the variables from the first part such that for $\alpha \neq \beta$, $\varphi|_{\rho_\alpha}$ and $\varphi|_{\rho_\beta}$ are different Boolean functions. Hence the paths in D corresponding to different ρ_α should end in different nodes. Hence, the size of D is at least 2^T.

We call a row of X as *special* if it contains at least one variable from Z. There are exactly T special rows in X. Notice that the rows i_0 and j_0 are not special, since all clauses containing variables of special rows should be in $\mathcal{C}$.

We consider two cases depending on whether all special rows lie in $\{s_1, \dots, s_{\lfloor (n-1)/2 \rfloor}\}$ or there exists at least one *special* row in $\{s_{\lfloor (n-1)/2 \rfloor + 1}, \dots, s_n\}$.

For convenience we denote $S_{i \dots j} = \{s_i, \dots, s_j\}$.

Case 1. All *special* rows are in $S_{1 \dots \lfloor (n-1)/2 \rfloor}$. We denote the set of special rows by $\{w_1, \dots, w_T\} \subset S_{1 \dots \lfloor (n-1)/2 \rfloor}$.

For every $\alpha \in \{0,1\}^T$ we define a substitution ρ_α into the variables of the first part.

- Let $k(i)$ denote the index of π-first variable of the w_ith row. ρ_α substitutes to the variables of X_{w_i} from the first part corresponding values of the binary string $a_i + e_{k(i)} \cdot \alpha_i$; recall that for $k \in [\ell]$, e_k denote the element of $\{0,1\}^l$ with $\ell - 1$ zeros and the only one on the kth place. We notice that the variable $X_{w_i, k(i)}$ is necessarily in the first path.
- If $i \in \{i_0, j_0\} \cap S_{1 \dots \lfloor (n-1)/2 \rfloor}$, then ρ_α substitutes to the variables from X_i in the first path corresponding values from the binary string a_0.
- Let $J_i := \{j \in [\ell] \mid X_{w_i, j} \in Z\}$. For every non-special row j from $S_{1 \dots (n-1)/2}$ we choose a unique string b_j such that $b_j \neq a_0$ and for all $i \in [k]$ there exists $r \in [\ell] \setminus (J_i \cup \{k(i)\})$ such that the rth bit of b_j differs from the rth bit of a_i. Since $|J_i| \leq d$ and $T 2^{d+1} + 1 < n/2$,

such strings indeed exist. We assume that the choice of b_j does not depend on α. For all $j \in S_{1\dots(n-1)/2}$, ρ_α substitutes to the variables from X_j in the first path corresponding values from b_j.

- Notice that rows from $S_{\lfloor (n-1)/2\rfloor+1\dots n-1}$ do not contain variables from the first part.

Now we show that for every $\alpha \neq \beta \in \{0,1\}^T$ functions $\varphi|_{\rho_\alpha}$ and $\varphi|_{\rho_\beta}$ are different. To do it we construct a substitution ρ to the variables of the second part such that ρ sets one of the functions to zero and the other to one. Since $\rho_\alpha \neq \rho_\beta$ then, without loss of generality, we assume that there exists row $m \in [T]$ such that $\alpha_m = 0$ and $\beta_m = 1$.

We define ρ as follows:

- For each *non-special* row $j \in S_{1\dots\lfloor(n-1)/2\rfloor}$, ρ substitutes to the second part variables from X_j values from b_j.
- For each *special* row w_i for $i \in [T]$, ρ substitutes to the second part variables from X_{w_i} values from a_i.
- For each row from $S_{\lfloor(n-1)/2\rfloor+1\dots\lfloor(n-1)/2\rfloor+2^{|J_m|}}$, ρ substitutes to the variable of the second part corresponding values of an element from $V(a_m, J_m) := \{s \in \{0,1\}^l \mid s \text{ agrees with } a_m \text{ on } [l] \setminus J_m\}$. Since both the sets $S_{\lfloor(n-1)/2\rfloor+1\dots\lfloor(n-1)/2\rfloor+2^{|J_m|}}$ and $V(a_m, J_m)$ have $2^{|J_m|}$ elements, we can assume that for different rows ρ uses different elements of $V(a_m, J_m)$.

The definition of ρ is not finished yet but we already can show the following.

▷ Claim 28. $\varphi|_{\rho_\alpha \cup \rho} = 0$

Proof. Let τ be the restriction of $\rho_\alpha \cup \rho$ to $X \setminus Z$. If we apply τ to X_{w_m} and then substitute any values into the variables of X_{w_m} from $\mathcal{Z}$, we obtain a string from $V(b_m, J_m)$. But all such strings are substituted by $\rho_\alpha \cup \rho$ into other rows of the variables matrix. Since $\mathcal{C}$ contains all the clauses of BPHP_n^{n+1} that forbid the row X_{w_m} to be equal to any other row, there **do not** exist values for variables from $\{X_{w_m,j} \mid j \in J_m\}$ such that all clauses of $\mathcal{C}$ are satisfied and hence $\varphi|_{\rho_i \cup \rho} = \varphi|_\tau = \exists z_1, \dots, z_b \bigwedge_{C \in \mathcal{C}} C|_\tau = 0$. Note that we heavily rely on the fact that all clauses containing variables from special rows are in $\mathcal{C}$. ◁

- Now we have to define ρ on other variables i.e. variables from the rows $S_{\lfloor(n-1)/2\rfloor+2^{|J_m|}+1\dots n-1}$ such that $\varphi|_{\rho_\beta \cup \rho} = 1$. Note that:
  - All strings substituted by $\rho_\beta \cup \rho$ to rows from $S_{\lfloor(n-1)/2\rfloor+1\dots\lfloor(n-1)/2\rfloor+1+2^{|J_k|}}$ are different.
  - Since $\{a_0, a_1, \dots, a_T\}$ is a code with distance at least $d+3$ and $|J_m| \leq d$, strings from $V(a_m, J_m)$ that are substituted by $\rho_\beta \cup \rho$ to rows $S_{\lfloor(n-1)/2\rfloor+1\dots\lfloor(n-1)/2\rfloor+1+2^{|J_k|}}$ differ from the strings substituted by $\rho_j \cup \rho$ to rows $\{w_j | j \in [T] \setminus m\}$ and a_0.
  - All strings from $V(a_m, J_m)$ are different from $a_m + e_{k(m)} \cdot \beta_m = a_m + e_{k(m)}$ in the $k(m)$th bit.
  - No string from $V(a_m, J_m)$ can be substituted by $\rho_\beta \cup \rho$ to non-special rows.

So far $\rho_\beta \cup \rho$ substitutes values to variables from several rows. Notice that every binary string except a_0 is used at most once and a_0 is used twice (for rows i_0 and j_0). First of all, we can extend $\rho_j \cup \rho$ to all variables such that we will have only two equals rows (i_0 and j_0 equals a_0). Hence, $\rho_\beta \cup \rho$ satisfies $\bigwedge_{C \in \mathcal{C}} C$, thus by Lemma 5, $\rho_\beta \cup \rho$ satisfies φ.

Case 2. There exists $w_{k_0} \in S_{\lfloor (n-1)/2 \rfloor +1 \dots n-1}$ for some $k_0 \in [T]$. We fix an arbitrary T non-special rows $f_1, f_2, \dots, f_T$ from $S_{1 \dots \lfloor (n-1)/2 \rfloor}$ (it can be done since $T < n/4$).

For every $\alpha \in \{0,1\}^T$ we define a substitution ρ_α to the variables of the first part:

- Let now $k(i)$ denote the index of π-first variable of the f_ith row. For every $i \in [T]$, ρ_α substitutes to the variables of X_{f_i} from the first part corresponding values of the binary string $a_i + e_{k(i)} \cdot \alpha_i$; recall that for $k \in [\ell]$, e_k denote the element of $\{0,1\}^l$ with $\ell - 1$ zeros and the only one on the kth place. We notice that the variable $X_{f_i, k(i)}$ is necessarily in the first part.
- If $i \in \{i_0, j_0\} \cap S_{1 \dots \lfloor (n-1)/2 \rfloor}$, then ρ_α substitutes to the variables from X_i in the first part corresponding values from the binary string a_0.
- We say that a binary string $s \in \{0,1\}^l$ is *bad* if $s = a_0$ or there exist $b \in \{0,1\}^l$ with the Hamming distance at most 1 from s and $i \in [T]$ such that b agrees with a_i on the set of bits $J_{k_0} \cup \{k(i)\}$. The number of bad strings is at most $1 + (l+1) \cdot T \cdot 2^{d+1} < n/2$. So for every row $i \in S_{1 \dots \lfloor (n-1)/2 \rfloor} \setminus \{f_1, \dots, f_T, i_0, j_0\}$ we can choose not bad string b_i by some fixed way that is not dependent on α. ρ_α substitutes to the first part variables from X_i corresponding values from b_i.

For every distinct $\alpha, \beta \in \{0,1\}^T$ we build a substitution ρ into the variables of the second part that separates $\varphi|_{\rho_\alpha}$ and $\varphi|_{\rho_\beta}$. Without loss of generality assume that there exists index $m \in [T]$ such that $\alpha_m = 0$ and $\beta_m = 1$.

We choose arbitrarily $2^{|J_{k_0}|} - 1$ non-special rows $g_1, \dots, g_{2^{|J_{k_0}|}-1}$ in $S_{\lfloor (n-1)/2 \rfloor \dots n-1}$. Recall that $k(m)$ is the index of the π-first variable of the row f_m. Consider two sub-cases:

Case 2.1. $k(m) \notin J_{k_0}$. Let us define a partial substitution ξ; the substitution ρ will be an extension of ξ.

- For every $i \in [T]$, ξ substitutes to the variables of X_{f_i} from the second part corresponding values of the binary string a_i
- If $i \in \{i_0, j_0\}$, then ξ substitutes to the variables from X_i in the second part corresponding values from the binary string a_0.
- For $i \in S_{1 \dots \lfloor (n-1)/2 \rfloor} \setminus \{f_1, \dots, f_T, i_0, j_0\}$, ξ substitutes to the second part variables from X_i corresponding values from b_i.
- ξ substitutes to variables from $X_{w_{k_0}}$ corresponding values from a_m.
- To the rows $g_1, \dots, g_{2^{|J_{k_0}|}-1}$, ξ substitutes distinct values from $V(a_m, J_{k_0}) \setminus \{a_m\}$. (Notice that both sets have the same cardinality $2^{|J_{k_0}|} - 1$).

▷ Claim 29. $\varphi|_{\rho_\alpha \cup \xi} = 0$.

Proof. The proof is similar to the proof of Claim 28. Recall that J_{k_0} are indices of Z-variables from w_{k_0}. Let τ be restriction of $\rho_\alpha \cup \xi$ to non-Z variables; τ substitutes to non-Z variables of w_{k_0} values from a_m. All strings from $V(a_m, J_{k_0})$ are already substituted by $\rho_\alpha \cup \xi$ to the rows $g_1, \dots, g_{2^{|J_{k_0}|}-1}$ and f_m. So, for every fixed values of Z-variables, $\rho_\alpha \cup \xi$ falsifies some clause from $\mathcal{C}$. Hence by the same reasons as in Claim 28, $\varphi|_\tau = \varphi|_{\rho_i \cup \xi} = 0$. ◁

Now we have to define ρ by extending ξ on other variables to ensure that $\varphi|_{\rho_\beta \cup \rho} = 1$. Since $k(m) \notin J_{k_0}$, $\rho_\beta \cup \xi$ does not substitutes an element of $V(a_m, J_{k_0})$ to X_{f_m}, hence all strings that are substituted by $\rho_\beta \cup \xi$ so far are distinct except a_0 that is substituted two times: to rows X_{i_0} and Y_{i_0}. But $\mathcal{C}$ does not forbid to have X_{i_0} and X_{j_0} with value a_0. So we can continue ξ to ρ to satisfy ϕ as we did in Case 1.

Case 2.2. $k(m) \in J_{k_0}$. We first try to define ρ as an extension of ξ as in Case 2.1:

By Claim 29, $\varphi|_{\rho_\alpha \cup \xi} = 0$. But now we have a problem with extending ξ to ρ such that $\varphi|_{\rho_\beta \cup \rho} = 1$ since in this case $\rho_\beta \cup \xi$ substitutes to X_{f_m} the string $a_m + e_{k(m)}$ from $V(a_m, J_{k_0})$ and, hence, $\rho_\beta \cup \xi$ substitutes the same string into two different rows of the matrix: f_k and some row from $\{g_1, \ldots, g_{2^{|J_{k_0}|}-1}\}$. If it is not forbidden by clauses from $\mathcal{C}$ then we can continue ξ to ρ such that $\varphi|_{\rho_j \cup \rho} = 1$ as in the previous case.

Otherwise, we redefine ξ in the following way: we flip the value of ξ on some non-Z variable from $X_{w_{k_0}}$. Now all strings that are substituted by $\rho_i \cup \xi$ are different (except a_0) since only a_m and a_0 were substituted two times by the old version of ξ. Here we use the fact that the new string substituted to w_{k_0} is bad since it is within distance 1 from a_m. Hence we can continue ξ to ρ such that $\varphi|_{\rho_\alpha \cup \rho} = 1$. But now $\varphi|_{\rho_\beta \cup \rho} = 0$ since two different non-special rows are substituted with $a_m + e_{k(m)}$ and it is forbidden by a clause from φ. ◀

5 Further Research

In this section, we would like to highlight a few open questions that naturally follow from our current research:

1. Prove a super-polynomial lower bound for the OBDD$(\wedge, \exists, \text{reordering})$ proof system.
2. Is BPHP_n^{n+1} hard for the OBDD$(\wedge, \exists)$ proof system?
3. Is it possible to separate the OBDD$(\wedge, \exists)$ proof system and OBDD$(\wedge, \exists)$ algorithms?

References

1 Alfonso San Miguel Aguirre and Moshe Y. Vardi. Random 3-SAT and BDDs: The plot thickens further. In *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, pages 121–136, 2001. doi:10.1007/3-540-45578-7_9.

2 Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *Journal of Automated Reasoning*, 35(1-3):51–72, 2005. doi:10.1007/s10817-005-9006-x.

3 Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, pages 77–91, 2004. doi:10.1007/978-3-540-30201-8_9.

4 Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986. doi:10.1109/TC.1986.1676819.

5 Sam Buss, Dmitry Itsykson, Alexander Knop, Artur Riazanov, and Dmitry Sokolov. Lower bounds on OBDD proofs with several orders. *ACM Trans. Comput. Log.*, 22(4):26:1–26:30, 2021. doi:10.1145/3468855.

6 Sam Buss, Dmitry Itsykson, Alexander Knop, and Dmitry Sokolov. Reordering rule makes OBDD proof systems stronger. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 16:1–16:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.CCC.2018.16.

7 Wěi Chén and Wenhui Zhang. A direct construction of polynomial-size OBDD proof of pigeon hole problem. *Information Processing Letters*, 109(10):472–477, 2009. doi:10.1016/j.ipl.2009.01.006.

8 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. URL: http://www.jstor.org/stable/2273702.

9 E. N. Gilbert. A comparison of signalling alphabets. *The Bell System Technical Journal*, 31(3):504–522, 1952. doi:10.1002/j.1538-7305.1952.tb01393.x.

10 Ludmila Glinskih and Dmitry Itsykson. Satisfiable Tseitin formulas are hard for nondeterministic read-once branching programs. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 26:1–26:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.MFCS.2017.26.

11 Ludmila Glinskih and Dmitry Itsykson. On tseitin formulas, read-once branching programs and treewidth. *Theory Comput. Syst.*, 65(3):613–633, 2021. doi:10.1007/S00224-020-10007-8.

12 Venkatesan Guruswami. List decoding from erasures: bounds and code constructions. *Institute of Electrical and Electronics Engineers. Transactions on Information Theory*, 49(11):2826–2833, 2003. doi:10.1109/tit.2003.815776.

13 R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x.

14 Dmitry Itsykson, Alexander Knop, Andrei E. Romashchenko, and Dmitry Sokolov. On OBDD-based algorithms and proof systems that dynamically change the order of variables. *J. Symb. Log.*, 85(2):632–670, 2020. doi:10.1017/JSL.2019.53.

15 Jan Krajícek. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *The Journal of Symbolic Logic*, 73(1):227–237, 2008. doi:10.2178/jsl/1208358751.

16 Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

17 Chunxiao Li, Noah Fleming, Marc Vinyals, Toniann Pitassi, and Vijay Ganesh. Towards a complexity-theoretic understanding of restarts in SAT solvers. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 233–249. Springer, 2020. doi:10.1007/978-3-030-51825-7_17.

18 Stefan Mengel. Bounds on BDD-based bucket elimination. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, pages 16:1–16:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SAT.2023.16.

19 Guoqiang Pan and Moshe Y. Vardi. Symbolic techniques in satisfiability solving. *Journal of Automated Reasoning*, 35(1-3):25–50, 2005. doi:10.1007/s10817-005-9009-7.

20 Nathan Segerlind. On the relative efficiency of resolution-like proofs and ordered binary decision diagram proofs. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 100–111. IEEE Computer Society, 2008. doi:10.1109/CCC.2008.34.

21 R. R. Varshamov. The evaluation of signals in codes with correction of errors. *Dokl. Akad. Nauk SSSR*, 117:739–741, 1957.

22 Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM, 2000.