On small-depth Frege proofs for PHP

Johan Håstad
dept. of Mathematics
KTH Royal Institute of Technology
Stockholm, Sweden
johanh@kth.se

***Abstract*—We study Frege proofs for the one-to-one graph Pigeon Hole Principle defined on the $n \times n$ grid where n is odd. We are interested in the case where each formula in the proof is a depth d formula in the basis given by $\wedge$, $\vee$, and $\neg$. We prove that in this situation the proof needs to be of size exponential in $n^{\Omega(1/d)}$. If we restrict the size of each line in the proof to be of size M then the number of lines needed is exponential in $n/(\log M)^{O(d)}$. The main technical component of the proofs is to design a new family of random restrictions and to prove the appropriate switching lemmas.**

***Index Terms*—complexity of proof procedures**

I. Introduction

In this paper we study formal proofs of formulas in Boolean variables encoding natural combinatorial principles. We can think of these as tautologies but it is often more convenient to think of them as contradictions. When a certain formula, F, is a contradiction then its negation, $\bar{F}$ is a tautology and we do not distinguish the two. In particular in the below discussion we might call something a tautology that the reader, possibly rightly, thinks of as a contradiction. We are equally liberal with usage of the word "proof" which might more accurately be called "a derivation of contradiction".

We are given a set of local constraints that we call "axioms". These are locally satisfiable but not globally in that there is no global assignment that satisfies all the axioms. A proof derives consequences of the axioms and it is complete when it reaches an obvious contradiction such as 1=0 or that an empty clause contains a true literal.

A key property of such a proof system is the kind of statements that can be used and in this paper we allow Boolean formulas over the basis $\wedge$, $\vee$, and $\neg$ where the alternation depth is d. Here d is a constant independent of the formula size or a slowly growing function of the size. A fundamental and popular case is resolution corresponding to $d = 1$, where each formula is a disjunction of literals.

It is far from easy to analyze resolution but this proof system has been studied for a long time and many questions are now resolved. We do not want to discuss the history of resolution but as it is very relevant for the current paper let us mention that an early milestone was obtained by Haken [Hak85] in 1985 when he proved that the Pigeon Hole Principle (PHP) requires exponential size resolution proofs. The PHP states that $m + 1$ pigeons can fly to m holes such that no two pigeons fly to the same hole. It has $(m + 1)m$ Boolean variables x_{ij} which is true iff pigeon i flies to the hole j. The axioms say that for each i there is a value of j such that x_{ij} is true and for each j there is at most one i such that x_{ij} is true. This is clearly a contradiction but to prove this counting is useful and resolution is not very efficient when it comes to counting.

The focus of this paper is the more powerful proof system obtained for larger values of d and here a pioneering result was obtained by Ajtai [Ajt94] proving superpolynomial lower bounds for the size of any proof for PHP for any fixed constant d. The lower bounds of Ajtai were not explicit and [BPU92] gave the first such bounds, namely that depth $\Omega(\log^* n)$ is needed for the size of the proof to be polynomial. This was greatly improved in two independent works by Krajíček, Pudlák, and Woods [KPW95] and Pitassi, Beame, and Impagliazzo [PBI93], respectively. These two papers established lower bounds for the size of any proof of the PHP of the form exponential in $n^{c^{-d}}$ where $c > 1$, and gave non-trivial bounds for depths as high as $\Theta(\log \log n)$.

Related questions were studied in circuit complexity where the central question is to study the size of a circuit needed to compute a particular function. Here a sequence of results [FSS84], [Sip83], [Yao85], [Hås86] established size lower bounds of the form exponential in $n^{\Omega(1/d)}$ and obtained strong lower bounds for d as large as $\Theta(\log n/ \log \log n)$. The results were obtained for the parity function and here it is easy to show that this function can be computed by circuits of matching size. To see that PHP allows proof of size exponential in $n^{O(1/d)}$ is more difficult but was established in 2001 by Atserias et al [AAG01].

A technique used in many of these papers is called "restrictions". The idea is simply to, in a more or less clever way, give values to most of the variables in the object under study and to analyze the effect. One must preserve[1] the function computed (or tautology being proved) while at the same time be able to simplify the circuits assumed to compute the function or the formulas in the claimed proof. An important reason that the lower bounds in circuit complexity were stronger than those in proof complexity is that it is easier to preserve a single function than an entire tautology with many axioms.

After being stuck at $d = O(\log \log n)$ for decades we have recently seen significant progress. The first step was taken by Pitassi et al [PRST16] who obtained super-polynomial lower bounds for depths up to any $o(\sqrt{\log n})$. The tautology

[1]One does not really preserve a function or a formula and an object of size n is reduced to a similar object of size $f(n)$ for some $f(n) < n$.

considered was first studied by Tseitin [Tse68] and considers a set linear equations modulo two defined by a graph. The underlying graph for [PRST16] is an expander. These results were later extended to depth almost logarithmic by Håstad [Hås20] and in this case the underlying graph is the two-dimensional grid. The bounds obtained were further improved by Håstad and Risse [HR22].

All results mentioned so far only discuss total size. For resolution each formula derived is a clause and hence of size at most n but for other proof systems it is interesting to study the number of lines in the proof and the sizes of lines separately. Pitassi, Ramakrishnan and Tan [PRT21] had the great insight that a technical strengthening of the used methods yields much stronger bounds for this measure than implied by the size bounds. They combined some of the techniques of [Hås20] with some methods from [PRST16] to establish that if each line is of size at most M then the number of lines in a proof that establishes the Tseitin principle over the grid needs to be exponential in $n2^{-d\sqrt{\log M}}$. By using some additional ideas Håstad and Risse [HR22] fully extended the techniques of [Hås20] to this setting improving the bounds to exponential in $n/(\log M)^{O(d)}$.

Despite this progress, the thirty year old question whether the PHP allows polynomial size proof of depth $O(\log\log n)$ remained open. The purpose of this paper is to prove that it does not and that lower bounds similar to those for the Tseitin tautology also apply to the PHP. To build on previous techniques we study what is known as the graph PHP where the underlying graph is an odd size two-dimensional grid.

As the side length of the grid is odd, if one colors it as a chess board, the corners are of the same color and let us assume this is white. In the graph PHP on the grid, there is a pigeon on each white square and it should fly to one of the adjacent black squares that define the holes. This graph PHP is the result of the general PHP where most variables are forced to be 0. Each pigeon is only given at most four alternatives. Clearly any proof for the general PHP can be modified to give a proof of the graph PHP. To limit ourselves further we prove our lower bounds for the one-to-one PHP where we also have the axioms that each hole receives exactly one pigeon.

Phrased slightly differently, the one-to-one PHP on the grid says that there is a perfect matching of the odd size grid and we heavily use local matchings. We can compare this to the Tseitin tautology on the grid studied by [Hås20], [PRT21], [HR22] that states that it is possible to assign Boolean values to the edges of the grid such that there is an odd number of true variables next to any node. As a perfect matching would immediately yield such an assignment, the PHP is a stronger statement and possibly easier to refute. In particular, the result of the current paper implies a result similar to [HR22], but with slightly weaker bounds. Our methods are quite similar to those of that paper so this is not a different proof. Let us discuss the main technical point, namely to prove a "switching lemma".

By assigning values to most variables in a formula it is possible to switch a small depth-two formula from being a CNF to being a DNF and the other way around. In the basic switching lemma used to prove circuit lower bounds [FSS84], [Yao85], [Hås86], uniformly random constant values are substituted for a majority of the variables. Such restrictions are the easiest to analyze but are less useful in proof complexity as they do not preserve any interesting tautology.

To preserve a tautology or a complicated function it is useful to replace several old variables by the same new variable, possibly negated. It is possible to be even more liberal and allow old variables to be replaced by slightly more complicated expressions in the new variable. This technique was first introduced explicitly by Rossman, Servedio, and Tan [RST15] when studying the depth hierarchy for small-depth circuits but had been used in more primitive form in earlier papers. We use such generalized restrictions in this paper.

The technical strengthening needed by [PRT21] that we discussed above is to improve the standard switching lemma to what is commonly known as a multi-switching lemma. This concept was first introduced independently by Håstad [Hås14] and Impagliazzo, Mathews, and Paturi [IMP12] to study the correlation of small-depths circuits and simple functions such as parity.

In this setting one considers many formulas $(F^i)_{i=1}^m$ and the goal is to switch them all simultaneously in the following sense. There is a small depth (common) decision tree such that at any leaf of the tree it is possible to represent each F^i by a small formula of the other type. It was the insight that multi-switching could be used in the proof complexity setting that made it possible for [PRT21] to derive the strong bounds on the number of lines in a proof when each line is short.

The techniques used in this paper offer no surprise. We introduce a new space of restrictions that preserves the grid one-to-one PHP and for which it is possible to prove a standard switching lemma and then extend it to a multi-switching lemma. The most novel part is to design this new space of restrictions. It has many similarities with the space introduced in [Hås20] and from a very high level point of view, the proofs follow the same path. At the more detailed level in this paper we work with partial matchings of the grid which is a more rigid object than assignment that only satisfy the Tseitin condition of an odd number of true variables next to any node. This results in considerable changes in the details and as a result we get slightly worse bounds.

Once the space of restrictions is in place, two tasks remain. Namely to prove the switching lemmas and then use these bounds to derive the claimed bounds on proof size. This latter part hardly changes and we do not even repeat all details here.

An outline of the paper is as follows. We start with some preliminaries and recall some facts from previous papers in Section II. We introduce our new space of random restrictions in Section III. The basic switching lemma is proved in Section IV and we use it to establish the lower bound for proof size in Section V. We give the multi-switch lemma in Section VI and use it, in Section VII, to derive the lower bounds on the number of lines in a proof. We end with some very brief comments in Section VIII.

II. Preliminaries

In this section we give some basic definitions and derive some simple properties. We also recall some useful facts from related papers.

A. The formula to refute

We study the one-to-one PHP on the odd size grid. Nodes are given indexed by (i, j) where $1 \leq i, j \leq n$ and a node is connected to other nodes where one of the two coordinates is the same and the other differ by 1. As opposed to some previous papers it is here important that we are on the grid and not on the torus as we want a bipartite graph. There is one variable for each edge of the grid and an axiom saying that exactly one of the four variables next to a node is true.

We assume that there is one more white node than black node and hence pigeons are white nodes and holes are black nodes. Locally, however, holes and pigeons are very similar.

B. Frege proofs

We consider proofs where each line in the proof is either an axiom or derived from previous lines. The derivation rules are not important and all we need is that they are of constant size and sound. We use the same rules as [PRST16], [Hås20], [PRT21], and [HR22]. We demand that each formula that appears is of depth at most d and, as several previous papers, we do not allow $\wedge$ and count the number of alternations of $\neg$ and $\vee$. The $\wedge$ operator simulated by $\neg \vee \neg$. The rules are as follows.

- (Excluded middle) $(p \vee \neg p)$
- (Expansion rule) $(p \to p \vee q)$
- (Contraction rule) $(p \vee p) \to p$
- (Association rule) $p \vee (q \vee r) \to (p \vee q) \vee r$
- (Cut rule) $p \vee q, \neg p \vee r \to q \vee r$.

The concept of t-evaluations was introduced by Krajíček et al. [KPW95] and is a very convenient tool for proving lower bounds on proof size. Here we follow the presentation of Urquhart and Fu [UF96] while using the notation of [Hås20] and [HR22]. A t-evaluation is a map from formulas to decision trees of depth at most t. It is important that values along any branch in such a decision tree are locally consistent and hence let us first look at decision trees.

C. Decision trees and t-evaluations

Normally a decision tree asks for values of variables but we instead allow only questions of the form

"To which node is i matched?"

Clearly the answer to this question determines the value of any variable next to i and thus is more powerful than a single variable question. On the other hand it can be simulated by asking ordinary variable questions for three variables around i. Thus within a factor of three in the number of questions, this type of questions is equivalent to variable queries.

We say that a decision tree is a 1-tree if all its leafs are labeled one and similarly we have 0-trees. It might seem redundant to allow such trees but when doing operations on decision trees such as taking the logical or of a number of trees, they naturally occur.

We maintain the property that values obtained along any branch in any decision tree are locally consistent.

Definition 2.1: The matching M of size t is *locally consistent* if it can be extended to a complete matching of a larger set $S \times T$. We require that each of S and T is the union of even size intervals such that the total length of all intervals in each of the sets is at most $2t$.

The reason for the above definition is that locally consistent matchings can always be extended to include additional vertices.

Lemma 2.2: Given a locally consistent matching, M of size at most $n/20 - 1$, and a node v not matched by M. It is possible to find a partner, w, of v, such that M jointly with (u, w) is a locally consistent matching.

Proof: If v is already in $S \times T$ we can use the same extension. Suppose $v = (a, b)$ where $a \in S$ and $b \notin T$. It is easy to find b' such that $T \cup \{b, b'\}$ is a union of even size intervals. Now we can add matchings of $S \times b$ and $S \times b'$ using that S is a union of even size intervals.

The case when $a \notin S$ and $b \in T$ is symmetric and let us handle the case $a \notin S$ and $b \notin T$. We can find b' as in the previous case enlarging T to $T' = T \cup \{b, b'\}$ and then proceed by adding a and a suitable a' to S. ∎

We are interested in collections of formulas $(F^i)_{i=1}^m$ and the simultaneous evaluation of these formulas. We say that these have an ℓ common decision tree of depth s if there is a single decision tree of depth s such that at any leaf of this decision tree, each F^i can be represented by a depth ℓ decision tree.

Remark. In the intuitive notion of "locally consistent" a natural property is that any sub-assignment of a locally consistent assignment is locally consistent. This is not obviously true in our definition as the sizes of S and T depend on the size of the matching. From now on we let the informal notion "locally consistent" be short for "formally locally consistent or extendable to a formally locally consistent assignment".

As stated above, a t-evaluation is a mapping, φ, of formulas to decision trees of depth at most t and we want it to have some properties.

1) The constant true is represented by a 1-tree and the constant 0 is represented by a 0-tree.
2) If F is an axiom of the PHP contradiction then $\varphi(F)$ is a 1-tree.
3) If $\varphi(F) = T$ then $\varphi(\neg F)$ is a decision tree with the same topology as T but where the value at each leaf is negated.
4) Suppose $F = \vee F_i$. Consider a leaf in $\varphi(F)$ and the assignment, τ leading to this leaf. If the leaf is labeled 0 then for each i $\varphi(F_i)\lceil_\tau$ is a 0-tree and if the leaf is labeled 1 then for some i, $\varphi(F_i)\lceil_\tau$ is a 1-tree.

The key property for t-evaluations is the following lemma.

Lemma 2.3: Suppose we have a derivation using the rules of Section II-B starting with the axioms of the one-to-one PHP on the $n \times n$ grid. Let Γ be the set of all sub-formulas

of this derivation and suppose there is a t-evaluation whose range includes Γ where $t \leq n/10$. Then each line in the derivation is mapped to a 1-tree. In particular we do not reach a contradiction.

Proof: We only sketch the proof as it is tedious and the essentially the same as the proof of the similar lemma in [Hås20]. We need two properties, namely that each axiom is represented by a 1-tree, and that the derivation rules preserve this property. The first property is true by definition. The second property follows from the fact that the derivation rules are sound and we never "get stuck" in a decision tree. By this we mean that it always possible to continue a branch in a decision tree keeping the values locally consistent. This is ensured by Lemma 2.2. ∎

When studying the number of lines in a proof where each line is short, an extension of Lemma 2.3 is needed. This was first done in [PRT21] and we rely on the argument in the full version of [HR22]. We do not repeat the argument here and just give a short summary. We refer to [HR22] for the full version.

The bottom line is that once we have a multi-switching lemma such as Lemma 6.1 below we can construct t-evaluations for each line of the proof separately. If these evaluations admit a t common decision tree of depth s and are consistent then it is sufficient to obtain a contradiction provided that $s + t \leq cn$ for a sufficiently small constant c.

III. Restrictions

As stated in the introduction we use a slightly more complicated object than a restriction which normally only gives values to some variables. A restriction in our setting fixes many variables to constants but also substitutes the same variable or its negation for some variables. In a few cases an old variable is substituted by a small logical formula which is a disjunction of size at most three.

We are given an instance of the PHP on the $n \times n$ grid and a restriction, for a suitable parameter T, reduces it to a smaller instance on the $(n/T) \times (n/T)$ grid.

We divide the grid in to $(n/T)^2$ squares, each with side length T. Inside each square there are Δ (for a parameter to be fixed) smaller squares that we from now on call "mini-squares". In the end we pick one mini-square inside each square and let these represent the smaller instance. Each square has a color in the natural way and exactly as the nodes in the original grid and the corner squares are white.

Between each mini-square, s_i and any mini-square s'_j in an adjacent square we have $3R$ (for a parameter to be chosen) edge disjoint paths, each of even length. We can map exactly all vertices on this path by matching each node of the path to the appropriate adjacent node. We are also interested in matching each node to the other neighbor on the path and in case we need to include one node in each of the mini-squares to which it is attached. We think of this as using the path as an augmenting path and hence we refer to these paths as "augmenting paths". A matching on such a path is of type 0 if it does not include any node from the attached mini-squares and otherwise it is of type 1. We call the node from the mini-square included in a type 1 path a "dent" in the mini-square. This possible dent is also called the point of attachment of the path.

We group the $3R$ paths in groups of three and within each group two attach at a node which is the same color as the color as the mini-square, while the third one attaches to a node of opposite color. Please note that as each path is of even length, the nodes of attachment are of different colors but so are the mini-squares to which the path attaches. Hence the attachment points are either both the same color as the respective mini-square or both the opposite color. The first two augmenting paths may be of type 0 or 1. The third path is always of type 1 and as these paths play little role in the argument we mostly ignore them from now on.

For each augmenting path P we have a corresponding Boolean variable x_P which indicates whether it is of type 0 or type 1. By the R fixed paths of type 1 we can conclude that for any mini-square, if exactly half of its varying paths are of type 0 (and hence the other half is type 1) then its central area has equally many white nodes as black nodes remaining.

We set up our restrictions such that it is uniquely determined by the values for the variables x_P. Outside the paths and and the mini-squares we more or less have a fixed matching and we give details below. Let us start by describing how to construct a matching inside a square with some dents on the perimeter.

A. Matching a square with dents

In this section we prove the following lemma.

Lemma 3.1: Suppose have square with even side length and which has the same number of white and black dents. Suppose further that there are at most M dents and no dent is within M of a corner. In this situation there is always a matching of the square. If we have a square with odd side length with one more white node, then the similar statement is true assuming we have one more white dent.

We do not explicitly give a bound for the side length of the square. It must be at least $2M$ to have any dent at distance M from all corners. On the other hand a side length of $4M$ certainly allows for M dents fulfilling the requirement.

Proof: Let us first do the case of even side length. Suppose the side length is S. Take any dent, and assume for concreteness that it is white. Start matching nodes along the perimeter starting with the node next to this dent. This is straightforward until we hit the next dent. If this dent is black we get a perfect match along the perimeter while if it is white we are forced to create a new white dent. We continue on the other side of this dent and go all around the square. We get a new square of side length $S - 2$ and some dents. A dent remains iff it is the same color as the dent preceding it.

As we have both black and white dents, the number of dents has decreased by at least two and the distance to the corner has decreased by at most 2. We repeat this until there are no remaining dents. The rest is easy to match.

If the side length is odd then the process stops when there is only one white dent on the boundary. Also in this case the remaining square is easy to match. ■

We use this lemma on mini-squares but also on small squares called bricks that we define below.

B. Details of mini-squares and paths

Let us specify some details of the construction. It is convenient to use the concept of *brick* which is a square of size $\alpha \times \alpha$ where α is a small even integer and the reader might think of it as 20. Thus a brick is a small even size sub-graph where, if it is not affected from the outside it is easy to find a matching.

All but one mini-square have side length $6\alpha\Delta R$. We think of its interior as one square where we can apply Lemma 3.1. To the exterior we have $6\Delta R$ bricks on each side and the interesting part is the middle $2\Delta R$ bricks. Half of these are used to route the $3\Delta R$ paths to the Δ mini-squares in the square in the given direction. The bricks next to the corners are only used to get the distance to the corners needed by Lemma 3.1.

We have a special single mini-square in the top left corner square[2]. It has side length $6\alpha\Delta R+1$ which in particular is an odd number but is otherwise like the other mini-squares. We call this the "designated survivor".

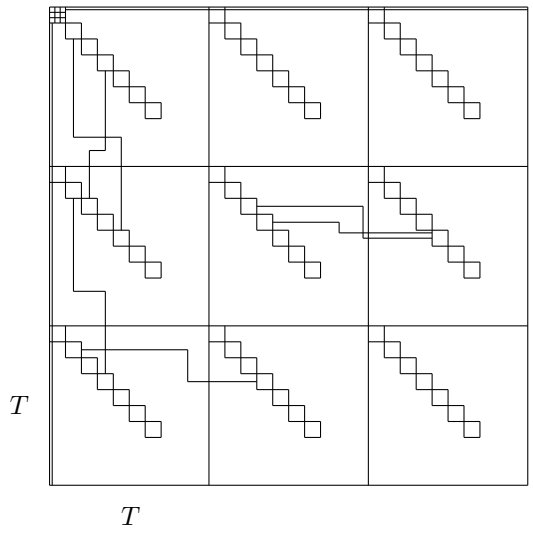

Fig. 1. The placement of mini-squares and squares and some paths. The designated survivor is the checkered mini-square. The matchings along top row and leftmost column are indicated by solid lines.

In the top row, the $n - 6\alpha R\Delta - 1$ nodes outside the designated survivor are matched in a horizontal matching. Similarly the nodes in the leftmost column outside the designated survivor are matched in a vertical matching. This basically eliminates one row and one column and we assume for convenience that $n \equiv 1$ modulo α and we cover the rest of the grid with bricks.

A square is a square of bricks of side length $7\Delta^2 R$ bricks and starting in the top left corner we have Δ mini-squares along the diagonal. This leaves $\Delta^2 R$ empty rows of bricks in the bottom of each square and $\Delta^2 R$ empty columns at the right. By our placement of the designated survivor and the elimination of the top row and first column also the top left square looks essentially the same as other squares. Let us describe how to route horizontal paths.

Fix a mini-square s_i in a square S and let us see how to route paths to mini-squares s'_j in the square to its right. For each pair (i, j) we reserve R columns, c^k_{ij}, $1 \leq k \leq R$, of bricks in the right part of S. This can be done has we have Δ^2 pairs of mini-squares and $\Delta^2 R$ empty columns of bricks.

For each j, we reserve R bricks, all with even index, on the right perimeter of s_i and the kth brick contains three paths that we route straight right to c^k_{ij}. Similarly we route paths straight left from the kth brick in the ith part of s'_j to the same column. This time using odd index bricks. The path is completed by using the suitable part of c^k_{ij} to connect the two pieces.

Connecting mini-squares vertically is done completely analogously and we omit the description.

It is easy to see that in a brick we either have no path going through, one path moving horizontally, one path moving vertically, one path making a bend, or one horizontal and one vertical path. A path has a point of attachment at the brick which is included in the path iff the path is of type 1. Making sure that colors of the points of attachment are the different entering and leaving a brick and that they are not too close to the corners (that is why we need to make α larger than four) we can appeal to Lemma 3.1. As long as we maintain the type of a path throughout it passage, we can always match the interior of each brick.

We summarize the properties we need from this construction.

Lemma 3.2: The values of the path variables x_P uniquely determines a matching on all paths and in any brick outside the mini-squares. In any mini-square except the designated survivor such that half of its adjacent x_P variables are true we can find a unique matching of the remainder of this mini-square. In the designated survivor we can, under the same condition match all but one node.

C. Almost complete matchings and restrictions

We let an *almost complete matching*, usually denoted τ, be an assignment to all variables x_P such that exactly half the variables next to any mini-square are true. Note that many such τ do exist and in particular we can pick half the paths in any group of $2R$ paths to be of the each type. There are many other ways to pick τ but we do not need this explicitly.

Let us describe how to pick a random restriction from our space. We make uniformly random choices but if some choice is very unlucky we start all over again. Let π_1 be a fixed matching of all squares except that of the designated survivor in the top left corner. We pick a restriction σ as follows.

[2]This could be any white square and choosing this particular square is just to make some fixed choice.

1) Pick a uniformly random τ respecting the condition that half the variables next to any mini-square are true. If any group of $2R$ variables between two fixed mini-squares has fewer than $R/2$ variables of either value, restart. We denote this event as "lopsided group".
2) Pick a random mini-square from each square, except the square of the designated survivor. Match these mini-squares according to π_1. Hopefully without causing confusion we keep the name π_1 for this matching of mini-squares. These nodes jointly with the designated survivor are the *chosen* mini-squares and we call this set U.
3) For each pair of mini-squares (s_1, s_2) matched in π_1 pick one augmenting path of type 1 and convert it to type 0. These are called chosen paths. The choice is based on an advice string B and discussed below.
4) For each pair of mini-squares (s_1, s_2) in U in adjacent squares but not matched in π_1 pick one augmenting path of type 0 and make it a chosen path. The choice of the path is based on the advice string. This is also done with s_1 being the designated survivor.

Let us see that this defines a smaller instance of PHP.

D. The reduced instance

The nodes of the new instance are given by the elements in U. They naturally form a $(n/T) \times (n/T)$ grid. We have chosen paths that are of type 0 that connect any two adjacent elements in U while for any other path P, the value of x_P is now fixed.

For each chosen path, P, we introduce a new variable, which we call y_P, indicating whether it should be switched to type 1. By Lemma 3.2 if exactly one variable on a path next to a mini-square in U is true, then it is possible to find a matching of this mini-square. Thus the local conditions of a mini-square turn in to an axiom of the new instance of PHP. Let us see how to replace the old variables in a supposed proof with these new variables.

Old variables not on edges in chosen mini-squares or in bricks with at least one chosen path are now fixed in a way respecting the corresponding axioms.

Consider a brick with at least one chosen path going through. There is only one chosen path between any two adjacent squares and it is not difficult to see that at most one chosen path goes through any brick. There are two possible matchings of this brick depending on the value of the corresponding variable y_P. If an edge e is present in neither we replace x_e by 0 and if it is present in both we replace it by 1. If e is present in only one we replace it by y_P or $\bar{y}_P$ in the natural way. It is easy to see that any axiom related to a node in the brick becomes true.

Similarly in a mini-square we have four (or fewer if it is on the perimeter) chosen paths next to it. These are controlled by four new variables that we here locally call y_i for $1 \leq i \leq 4$. The new local axiom is that exactly one of these four variables is true.

We have four different matchings of the mini-square depending on which y_i is chosen to be true. Look at an edge, e and suppose it appears in the matchings corresponding to y_2 and y_3 being true. In such a case we replace x_e by $y_2 \vee y_3$ and similarly in other cases. If e is in none of the four matchings we replace x_e by the constant 0 and if it is in all four we replace it by the constant 1.

It is easy to check that any original axiom inside the mini-square either reduces to true or that exactly one of the four y_i is true. Indeed looking at the disjunctions replacing the four variables x_e around any node, each y_i appears in exactly one. We summarize the discussion of this section as follows.

Lemma 3.3: A restriction σ reduces an axiom in the PHP in the grid of size $n \times n$ either to the constant true or an axiom in the PHP of the new variables y_P. Each variable x_e is substituted by a conjunction of up to three literals.

We do make a composition of restrictions and let us note that even after a sequence of compositions and original variable is only the conjunction of three variables in the restricted PHP. This follows from the fact that any local area is only affected by the local variables in the PHP.

As in previous papers these full restrictions are not used in the main argument and we work with partial restrictions that we now turn to.

E. Partial restrictions

A full restriction is a very rigid object with exactly one live mini-square in each square and we proceed to make a more random looking object.

Let k be a parameter equal to $C \log n (n/T)^2$ for a sufficiently large constant C. After we have completed the construction of σ we add the following steps.

1) Pick, without replacement of mini-squares, k uniformly random pairs of mini-squares in adjacent squares. This yields a matching π_2. If this pick is unbalanced as defined below redo this step.
2) Change the type of one augmenting path between any pairs of nodes in π_2 from 1 to 0. The choice of which of the at least $R/2$ paths of type 1 is based on the advice string and discussed below.

As we pick $2k$ mini-squares we expect roughly[3] $2C \log n$ live mini-squares in any square. If this number is larger than $4C \log n$ for any square then we consider the pick unbalanced and we restart. For any two adjacent squares S_1 and S_2 we expect $C \log n/2$ pairs (s_1, s_2) picked such that $s_i \in S_i$. If this number is smaller than $C \log n/4$ for any pair (S_1, S_2) we consider this unbalanced and restart.

We call the resulting restriction ρ. The mini-squares picked by π_1 and π_2 jointly with the designated survivor are called "live". Note that for any mini-square that is not live we have fixed the matching in this mini-square permanently. Outside

[3]The reason this is not exactly true is squares at the perimeter have only two or three neighboring squares. Such squares are less likely to have many live mini-squares. This results in a factor $(1 + o(1))$ more mini-squares in other squares but this small factor does not matter and we ignore it.

the mini-squares the matching is also fixed except in bricks with at least one live path going through.

F. Changing types of augmenting paths

In the above procedure, in two places we need to select an augmenting path and (possibly) change its type. This happens when changing a path from type 1 to type 0 because its end-points are matched in π_1 or π_2 and when opening up for changing the type from 0 to 1 by making it a chosen path.

We could accept to make this choice arbitrary by losing some factors of R in our bounds, but as it is always nice to avoid unnecessary loss let us describe a more efficient choice.

The choices of the $2R$ variables in a group corresponds to a vector in $\{0,1\}^{2R}$ and we want to modify one coordinate in order to change the Hamming weight from t to either $t+1$ or $t-1$ and let us suppose the latter. We want the choice to be limited and as invertible as possible. As the number of strings of weights t and $t-1$ are different we cannot achieve perfect invertability. Suppose first that $t \leq R$.

Definition 3.4: Suppose $t \leq R$. A mapping f mapping $\binom{2R}{t}$ to $\binom{2R}{t-1}$ and which maps each set to a subset, is a *k-almost bijection* if it is surjective and $|f^{-1}(x)| \leq k$ for any x.

The following below lemma is probably well known but as the proof is not difficult we prove it. It is likely that 4 can be improved to 3 but this does not matter greatly for us, as this only affects unspecified constants.

Lemma 3.5: If $R/2 \leq t \leq R$ then there is a 4-almost bijection.

Proof: Consider a bipartite graph where the the left hand side elements are subsets of size $t-1$ and the right hand side elements are subsets of size t. Connect two sets iff one is a subset of the other. It is well known (see for instance Corollary 2.4 in [Bol86]) that this graph has a matching, M_1, of size $\binom{2R}{t-1}$.

Modify the construction by making three copies of each left hand side node. Each copy is again connected to any set that contains it. It follows by the LYM inequality (stated as Theorem 3.3 in [Bol86]) that this graph has a matching M_2 of size $\binom{2R}{t}$.

Now define $f(x)$ as follows. If x is matched in M_1 let it be its partner in this matching. If x is not matched in M_1 define $f(x)$ to be the partner under M_2.

Due to the first condition f is onto. The property that $|f^{-1}(y)| \leq 4$ for follows as a preimage of y is either its partner under M_1 or a partner of one of its three copies under M_2. ∎

Taking the complement of both input and output we define a 4-almost bijection, g mapping $\binom{2R}{t}$ to $\binom{2R}{t+1}$ for $R \leq t \leq 3R/4$. We use f and g to guide our choices and in addition we have two bits of advice for each group.

When we want to convert a path from type 1 to type 0 between s_i and s'_j we look the types of all paths between the two mini-squares. This is a vector, v, in $\{0,1\}^{2R}$ which by the non-lopsidedness has Hamming weight t which is in the interval $[R/2, 3R/2]$. If $t > R$ we look at $g^{-1}(v)$ and consider the two bits of advice, b_1 and b_2. All we need to do is to ensure that each choice in $g^{-1}(v)$ is possible but to be explicit we can proceed as follows.

- If $g^{-1}(v)$ is of size one we pick the unique element.
- If $g^{-1}(v)$ is of size two we use b_1 to make the choice.
- If $g^{-1}(v)$ is of size three then $b_1 = b_2$ we pick the lexicographically first path and otherwise we use b_1 to choose between the other two paths.
- If $g^{-1}(v)$ is of size four then use the pair (b_1, b_2) to make the choice.

The reason for the advice string is to get a pure counting argument when we later analyze probabilities. It would have worked to pick a random element from $g^{-1}(v)$.

To make the situation uniform we have two advice bits for any pair of mini-squares in adjacent squares. Thus most of these bits are never used. We let B denote the values of all these bits.

If $t \leq R$ we instead consider $f(v)$ and change the type of the corresponding path. Finally if we want to allow to change the weight from t to $t+1$ we reverse the two cases.

G. Analyzing the probability of a restart

We make a restart either because of a lopsided group or an unbalanced pick of π_2 and we analyze these separately. We start with lopsided groups.

Lemma 3.6: The probability that uniformly random τ has lopsided group is $O(n^2 2^{-cR})$ for a positive constant c.

Based on this lemma we fix R to be $C \log n$ for a sufficiently large constant C such that the probability of having a lopsided group is $o(1)$. Let us prove Lemma 3.6.

Proof: The almost complete matching τ is defined by the variables x_P which we in this section choose to take values 1 and -1. For a uniformly random assignment to all variables, let Z be the vector of all mini-square sums. Let us denote the number of mini-squares by d making Z an integer vector of length d. When constructing τ we are conditioning on the event $Z = 0^d$.

Fix any two mini-squares s_1 and s_2 and let g be the group of $2R$ variables associated with paths between s_1 and s_2. Let X_g be sum of the variables in this group. We want to estimate the probability that $X_g = x$ where x is either at least R or at most $-R$. Let Z'_g be the set of mini-square sums when the paths between s_1 and s_2 are removed. As these two mini-squares also have other adjacent augmenting paths this is still a vector length d. Let v_x be the vector of length d that has x at positions s_1 and s_2 and is otherwise 0. We want to estimate

$$Pr[X_g = x \mid Z = 0^d] = Pr[X_g = x \wedge Z = 0^d]/Pr[Z = 0^d]$$

which equals

$$Pr[X_g = x \wedge Z'_g = -v_x]/Pr[Z = 0^d]$$

and as the two events are independent this equals

$$Pr[X_g = x]Pr[Z'_g = -v_x]/Pr[Z = 0^d].$$

We have the following lemma of which we postpone the proof.

Lemma 3.7: For any outcome $v \in \mathbb{Z}^d$ we have $Pr[Z'_g = v] \leq Pr[Z'_g = 0^d]$. The similar statement is true for any multi-graph where each edge appear an even number of times.

In view of the lemma we get the upper bound

$$Pr[X_g = x]Pr[Z'_g = 0^d]/Pr[Z = 0^d]$$

for the probability we want to estimate. Clearly $Pr[Z = 0^d] \geq Pr[X_g = 0]Pr[Z'_g = 0^d]$ and substituting this into the equation we get the upper bound $Pr[X_g = x]/Pr[X_g = 0]$. When $|x| \geq R$ this probability is 2^{-cR} for some explicit c and since there are at most n^2 pairs of mini-squares the lemma follows. ∎

Let us prove Lemma 3.7.

Proof: Let $f(v)$ be the probability that $Z'_g = v$. As v is the vector sum of contributions of single x_P, $f(v)$ is a giant convolution. A typical term, f_0, is a probability distribution that give weight $\frac{1}{2}$ to the vector v_1 and weight $\frac{1}{2}$ to the vector $-v_1$ where v_1 is similar to the vector v_x in the above proof. Consider the corresponding Fourier expansion

$$\hat{f}_0(x) = \sum_v f_0(v)e^{2\pi i(v,x)}$$

where (v, x) is the inner product of v and x and x belongs to the d-dimensional torus. As f_0 is symmetric under negation, $\hat{f}_0$ is real-valued. Summing all variables in a given group gives a distribution function which is a $2R$-fold convolution of f_0. Its Fourier expansion represents $\hat{f}_0^{2R}$ which is a positive function. Summing the contribution from all edges corresponds, on the Fourier side, of taking the product. We conclude that

$$\hat{f}(x) = \sum_v f(v)e^{2\pi i(v,x)}$$

is a real-valued positive function. The largest Fourier coefficients of such a function is the constant coefficient and this is exactly what we wanted to prove. ∎

Let us next discuss the balance condition when picking π_2.

Lemma 3.8: The probability that π_2 is unbalanced is $O(n^{-2})$ provided $C > C_0$ for some fixed constant C_0.

Proof: As this is a very standard argument let us only sketch it. If we picked the pairs of mini-squares with replacement the lemma would be completely standard. Let us analyze the dynamic process. To see that we do not pick more than $4C \log n$ mini-squares in any square with high probability we note two facts.

- As we only pick an $o(1)$ fraction of all mini-squares, at each point in time a fraction $(1 - o(1)$ of all pairs are available.
- In view of this the probability that any single mini-square is picked is only a $(1 + o(1))$ factor larger compared to the procedure with replacement.

That we are unlikely to pick many mini-squares in a single square now follows from the corresponding result for the process of picking with replacement.

We turn to the condition that we have at least $C \log n/4$ pairs in any two adjacent squares. Also this analysis is completely standard if edges are picked with replacement. If we condition on not picking more than $4C \log n$ mini-squares in any square the probability that a picked edge is between two given squares does not decrease by more than a factor $1-o(1)$. Hence the probability of picking very few edges between two given squares in the process without replacement is not so different compared to the probability of the same event in the process with replacement. We leave it to the reader to fill in the details. ∎

IV. The switching lemma

In this section we establish the following basic switching lemma.

Lemma 4.1: There is a constant A such that the following holds. Suppose there is a t-evaluation that includes $F_i, 1 \leq i \leq m$ in its range and let $F = \vee_{i=1}^m F_i$. Let σ be a random restriction from the space of restrictions defined in Section III. Then the probability that $F\lceil_\sigma$ cannot be represented by a decision tree of depth at most $2s$ is at most

$$\Delta(A(\log n)^3 t\Delta^{-1})^s.$$

Proof: We are interested in a σ that gives a long path in the decision tree. As in previous papers [Hås20], [HR22] we explore the canonical decision tree under the partial restriction ρ which we from now on call simply "a restriction" dropping the word "partial".

The restriction ρ determines the values of many variables. In fact values are unknown only in central areas of the live mini-squares and in bricks that contain an augmenting path between two live mini-squares.

For any variable x_e we define its influential mini-square(s). This is either a single mini-squares or two mini-squares. We want the property that if we know the values of all x_P around the influential mini-squares then this uniquely determines the value of x_e. If e is within a mini-square then this mini-square is its influential mini-square. If e is in a brick outside the mini-squares then the influential mini-squares are the closest end point(s) of the live path(s) in this brick. We do not consider the value of x_e known unless we completely know the situation at its influential mini-square(s). Such information is supplied by something we call matched pairs.

Definition 4.2: A *matched pair* (s_1, s_2) is the information that s_1 and s_2 should be matched by changing the type of an augmenting path from 0 to 1 between the two mini-squares.

Once we have the pair to match we need to select which augmenting path to change. We keep this information implicit. In the case of π_2 this is the path that it type changed when going from τ to ρ and in the case of a pair of chosen mini-squares it is the identity of the chosen path.

The above discussion says that if we want to know the value of x_e we look at its influential mini-squares. If there is no live influential mini-square its value is determined and otherwise we need additional information in the form of a matched pair containing the influential mini-square.

We now proceed to define the canonical decision tree. The process is guided by ρ and a set I of matched pairs. The

support of the set I is the mini-squares appearing in any matched pair and this is initially empty.

Letting T_i denote $\varphi(F_i)$, we go over the branches of T_i for increasing values of i and a fixed order for each tree. The *forceable* branch is the first branch leading to a one that can be followed given the current I and ρ. Let us be formal.

Before stage j we have information set I^j in the form of some matched pairs. It contains some pairs from π_2 and some pairs based on answers in the decision tree. In stage j we have forcing information J_j that makes us follow the forceable branch and it contains.

1) A set of matched pairs from π_2.
2) A set of matched pairs of chosen mini-squares, compatible with the information set I^j.

By "compatible" we mean that the resulting partial matching on chosen mini-squares is locally consistent in the sense of Definition 2.1.

For any chosen mini-square mentioned in J_j we ask for its partner in the decision tree. This information, jointly with the matched pairs from π_2 in J_j, forms the jth information set, I_j, and we set $I^{j+1} = I^j \cup I_j$.

Given I^{j+1} we can determine whether the forceable branch is followed. If it is, we answer 1 in the canonical decision tree and halt the process. Otherwise we go to the next stage and look for the next forceable branch. If there is no more forceable branch we halt with answer 0. Let T be the resulting decision tree. To see that this is a an acceptable choice for $\varphi(F)$, we have a pair of lemmas.

Lemma 4.3: Let γ be a set of answers in the decision tree. If there is no forceable branch given this information, then, for each i, $T_i\lceil_{\sigma\gamma}$ is a 0-tree.

Proof: Suppose there is a locally consistent branch in $T_i\lceil_{\sigma\gamma}$ that leads to a leaf labeled one. The information used to follow this branch can be used as forcing information. ■

Lemma 4.4: Let γ be a set of answers in the decision tree. If we answer 1 then there is an i such that $T_i\lceil_{\sigma\gamma}$ is a 1-tree.

Proof: In fact for T_i used for the construction of the forceable path we have now reached a leaf that is labeled 1. ■

We use Razborov's labeling argument [Raz95] to analyze the probability that we make $2s$ queries in the canonical decision tree. Take any branch of this length and suppose it was constructed during g stages using information sets J_j. Let $J^* = \cup_{j=1}^{g} J_j$ and as any query is a result of including an element in J^* we know that it contains at least s pairs and for notational convenience we assume that this number is exactly s. We proceed to analyze the probability that the process results in a J^* of this size. Let us start with an easy observation.

Lemma 4.5: The support sets of J_j are disjoint. The support of J_j is also disjoint with the support of I_i as long as $i \neq j$.

Proof: The parts coming from π_2 are clearly disjoint as π_2 is a matching. The pairs of chosen nodes are also disjoint as any mini-square included in J_j is included in I_j and later $J_{j'}$ are disjoint from I_j. ■

In the spirit of [Hås20] and [HR22] we want to find a restriction ρ^* with fewer live centers and then show how to reconstruct ρ from ρ^* and some external information. As we need to be careful with counting we construct a quadruple $\tau^*, U^*, \pi_2^*, B^*$ yielding ρ^* and use this together with external information to reconstruct τ, U, π_2, B that was used to define ρ.

Define ρ^* to be ρ joint with the information set J^*. In other words for a matched pair in J^* we change the type of one augmenting path between the two mini-squares and now consider the two endpoints to be dead. We proceed to find a quadruple that gives this restriction.

Initially we let π_2^* be π_2 with all matched pairs in J^* removed and $U^* = U$. For any edge in J^* between two chosen mini-squares s_1 and s_2 remove these two mini-squares from U^*. If there is a pair (s_1', s_2') in π_2^* where s_i' is in the same square as s_i then remove this pair from π_2^* and put s_1' and s_2' in to U^*. If there is no such edge we allow for two "holes" of two empty squares in U^*.

After we have constructed U^* and π_2^* we construct τ^* and B^* to complete the quadruple. Initially set τ^* to equal ρ^* and B^* to equal B.

We use the fixed matching of squares that defined π_1 to create matching π_1^* on U^*. Consider pairs of mini-squares matched in π_1^* or π_2^*. Any pair matched in π_2^* is also matched in π_2 and most pairs in π_1^* come from π_1 and some might come from π_2. In such a case we restore the augmenting path used to go from τ to ρ to its original type. We also keep the same values of the corresponding advice bits.

For pairs in π_1 not matched in π_1 or π_2 we find some possible preimage in the form of one augmenting path to change and some advice bits. By the original τ not being lopsided and the choice of how to use the advice bits this is always possible. This gives almost an almost complete matching τ^* and a set of advice bits B^* such that the tuple τ^*, U^*, π_2^* and B^* produces ρ^*.

The reason that is not quite an almost complete matching is due to holes in U^* and the square of the designated survivor. If the designated survivor was included in J^* then the sum of the variables around this mini-square is +2 and the sum around the mini-square that replaced it is -2. The sum around any hole and any unmatched mini-center in U^* (due to hole) is also -2.

We continue to define a process that, using external information, reconstructs τ, U, π_2 and B from τ^*, U^*, π_2^* and B^* and the trees T_i.

The important process is to reconstruct all forceable branches used in the construction of the canonical decision tree. The collection of information sets I_i and J_j might not all be consistent and we need the concept of a signature to make sure that this does not confuse the reconstruction process.

Definition 4.6: The signature of a live center determines whether it is chosen and in such a case which direction it is matched in its forceable branch.

The reconstruction is now as follows. We start with ρ^* and we reconstruct I_i and J_i in order. We let ρ_j^* be the restriction

obtained from ρ jointly with I_i for $i < j$ and J_i for $i \geq j$. Thus ρ_1^* is simply ρ^* which is the starting point. We have a set E of prematurely found chosen mini-squares jointly with their signatures. This is initially the empty set. We proceed as follows.

1) Find the next branch forced to one in any of the T_i by ρ_j^*.
2) Find, if any, mini-square of E whose information is used to follow this path. If this information would not have been consistent with I^{j-1} go to the next branch.
3) Read a bit to determine whether there are more live variables to be found on this branch. In such a case, read a number in $[t]$ to determine its position. If this variable has two influential mini-squares read another two bits to determine which of these two mini-squares are alive. For any alive influential mini-square, retrieve the corresponding signature(s) from external information and, if chosen, include the mini-squares in E. Go to step 2.
4) Reconstruct I_j and J_j. Details below.
5) Remove any chosen center included in J_j from E.

We need a lemma.

Lemma 4.7: If the information in E is consistent with the information in I^{j-1} then we found have the forceable branch.

Proof: Suppose $v \in E$. As it is in E it is a chosen mini-square and was included in $J_{j'}$ for some $j' \geq j$. As the current path is forced to one it must have been a potential forceable branch. If it was not the actual forceable branch it must be that at least one of the matched pairs needed was not allowed. Forcing information from π_2 is always allowed and thus the only problem could have been consistency on the chosen mini-squares. If none of the elements used on the current path gives any conflict with I^{j-1}, it was allowed and hence we must have the forceable branch. ∎

We need to discuss how to reconstruct I_j and J_j and start with the latter. For each matched pair in J_j we have recovered at least one end-point. If needed we use external information to recover the other end-point. This costs at most Δ. Whenever we recover a pair of adjacent mini-squares we use external information to recover the advice bits. This only costs 2 bits and thus we below focus on identifying mini-squares and do not mention the reconstruction of advice bits.

For each element in J_j we need to discover whether it is chosen (this is $O(1)$ external information) and to which node is is matched in I_j. If it is not chosen then the information in I_j is the same as the one in J_j. If it is chosen then the partner is either the same as in J_j and we are done or it is live in ρ_j^*. In the latter case, at cost $2C \log n$, we can find its identity. Once we have identified the two partners in a matching, at cost $O(1)$ we can reconstruct which augmenting path was used.

We reconstruct I_j and J_j and compute ρ_{j+1}^* and proceed to the next stage. At the end of this process we have reconstructed ρ and in the process we have also identified all the sets I_j and J_j and we know which pairs in J_j are chosen and which belong to π_2. For the pairs of chosen mini-squares we put them back in to U and any replacements are moved back to π_2. At cost $O(1)$ we can identify the advice bits of all mini-squares involved in a single move. This way we recover U, the full π_2, and B. Once we have these we can read off the original τ.

Let us calculate how much information was used. For each edge in J^* at least one end-point is discovered at cost at most t and the other at cost at most Δ. On top of this we have information such as the whether it is chosen and the signature at cost $O(1)$. This gives a total cost of at most $2^{O(s)}(t\Delta)^s$. In reconstructing I_i we might incur an additional cost of $O(\log n)$ per mini-square for an additional factor of $2^{O(s)}(\log n)^{2s}$ and finally a factor $2^{O(s)}$ to get advice bits for nodes that move out and into U. Thus the total cost for the external information is bounded by $2^{O(s)}(\log n)^{2s}(t\Delta)^s$.

We proceed to compare the number of quadruples τ, U, π_2, B to the number of quadruples $\tau^*, U^*, \pi_2^*, B^*$, and let us first assume that there are no holes. Both U and U^* contain one mini-square from each square but there is a difference that U contains the designated survivor while U^* might contain a different mini-square from this square. Thus the number of possibilities for U^* is a factor Δ larger than the number of possibilities for U.

Both τ and τ^* are essentially almost complete matchings. We do not allow τ to be lopsided but this only reduces the number of possibilities, by Lemma 3.6, by a factor $1 + o(1)$. It is the case that τ^* is only slightly more general but we only need an upper bound on the number of possibilities. On the other hand if U^* does not contain the designated survivor there are two mini-squares in τ^* where the sum is not zero. This is the designated survivor and its replacement in U^* in the same square. The number of τ^* with these fixed sums is, however, smaller by Lemma 3.7. There is no difference between B and B^* and thus the number of alternatives is the same. We conclude that if we denote the number of different τ, U and B by N, then the number of triples τ^*, U^*, and B^* is bounded by $\Delta N(1 + o(1))$.

The big difference is in the number of matchings π_2 and π_2^*. The former contains k edges and the latter only $k - s$. The former is also not unbalanced, but as this, by Lemma 3.8, only changes the number of possibilities by a factor $(1 + o(1))$ we from now on ignore this condition. Let $T_{k'}$ be the number of matchings of k' pairs of mini-squares picked without repeated mini-squares. The non-replacement makes it difficult to count exactly but we only need an approximate number and this is easier.

Lemma 4.8: For $k' \leq (n/T)^2\Delta/4$ we have $T_{k'-1} \leq T_{k'}k'\Delta^{-2}(n/T)^{-2}$.

Proof: Let us make a bipartite graph where on one side we have matchings of size k' and the other side matchings of size $k' - 1$. There is an edge between two matchings iff the smaller matching is a subset of the larger one. The degree on the k'-side is clearly k'.

Now take any matching of size $k'-1$. This involves $2(k'-1)$ mini-squares. As this is fewer than a quarter of all mini-squares, at most half of all edges are disqualified from being added due to intersecting with an existing edge. We have

$\Delta(n/T)^2$ mini-squares to start with and each has at least 2Δ possible partners. We conclude that there are at least $\Delta^2(n/T)^2$ edges to add and hence this is the minimal degree on the $k'-1$-side. Double-counting edges we conclude that

$$k'T_{k'} \geq \Delta^2(n/T)^2 T_{k'-1}$$

and this proves the lemma. ∎

By a repeated application of the above lemma and using $k = \Theta(\log n(n/T)^2)$ we conclude that that $T_{k-s} \leq 2^{O(s)}(\log n)^s \Delta^{-2s} T_k$. Recalling that we have $2^{O(s)}(\Delta t)^s(\log n)^{2s}$ possibilities for the external information, and $\Delta N(1+o(1))$ possibilities for U^*, τ^*, B we conclude that, ignoring the $(1+o(1))$ factors, that the number of quadruples reconstructed is

$$2^{O(s)}(\Delta t)^s(\log n)^{2s}\Delta N(\log n)^s\Delta^{-2s}T_k$$

which by collecting factors equals

$$2^{O(s)}(t(\log n)^3\Delta^{-1})^s\Delta N T_k.$$

As the total number of quadruples is NT_k, this completes the proof of Lemma 4.1 in the case when there are no holes.

If we allow r holes then the number of possibilities of U^* increases by a factor at most n^{2r}. The number of τ^* is different as some sums are no longer 0, but the sum at each mini-square is uniquely determined by U^*. Once this is fixed the number of τ^* is at most the number of almost complete matchings by Lemma 3.7.

For there to be a hole, by the balance condition, all the at least $C\log n/8$ edges between two squares must be present in J^*. This implies that $r = O(s/\log n)$ and the factor $n^{O(r)}$ can be absorbed in the factor $2^{O(s)}$. This completes the proof also in this case. ∎

V. The lower bound for proof size

In this section we establish one of the two main theorems of the paper.

Theorem 5.1: Assume that $d \leq O(\frac{\log n}{\log\log n})$ and let n be an odd integer. Then any depth-d Frege proof of the one-to-one PHP on the $n \times n$ grid requires size

$$\exp(\Omega(n^{1/(2d-1)}(\log n)^{O(1)})).$$

Proof: The proof follows the standard path. We use a sequence of restrictions and after the ith restriction any sub-formula of the proof of original depth at most i is in the range of the t-evaluation.

In particular after d restrictions, any sub-formula of the proof is in the range of the t-evaluation. By Lemma 3.3 what remains is a smaller one-to-one PHP instance and by Lemma 2.3, provided the size of the remaining grid is significantly larger than t, the proof cannot derive contradiction. We just need to specify the parameters to make sure that the switching lemma can be applied to all formulas simultaneously.

If the proof is of size 2^S and we ensure that the base of the exponential is at most $\frac{1}{4}$ in each iteration, we apply the switching lemma with $s = \Theta(S)$. For the first iteration we have $t = 1$ while for later iterations $t = s$.

To get the base of the exponential small it is sufficient to set $\Delta = \Omega((\log n)^3)$ in the first iteration and $\Delta = \Omega((\log n)^3 S)$ in later operations. Using that $T = \Theta(\Delta^2 R)$ we get that after d iterations the side length of the grid has shrunk from n to

$$n2^{-O(d)}(\log n)^{-7d}S^{2-2d}.$$

Provided that this is larger than $10s$ we get a contradiction and we this is true if

$$S^{2d-1} \leq cn2^{-O(d)}(\log n)^{-7d}$$

for a sufficiently small constant c. This concludes the proof of the theorem. ∎

Next we turn to the multi-switching lemma.

VI. Multi-switching

The extension to multi-switching only requires minor modifications and we start by stating the lemma.

Lemma 6.1: There is a constants A such that the following holds. Consider formulas F_i^m, for $m \in [M]$ and $i \in [n_m]$, each associated with a decision tree of depth at most t and let $F^m = \vee_{i=1}^{n_m} F_i^m$. Let σ be a random full restriction from the space of restrictions defined in Section III. Then the probability that $(F^m)_{m=1}^M$ cannot be represented by an ℓ common partial decision tree of depth at most $4s$ is at most

$$\Delta M^{4s/\ell}\big(A(\log n)^7 t\Delta^{-1}\big)^s.$$

Proof: We need to construct an ℓ common decision tree and we treat the formulas for increasing values of m. We have counter j, starting at 0, indicating the number of times we have found a long branch in a decision tree. We have an information set I^+ which initially is empty.

If a formula F^m admits a decision tree of depth at most ℓ under ρ and the current I^+ then there we proceed to the next formula. If not we set $m_j = m$ and execute the process of forming a canonical decision tree. Its discovers a branch of length at least ℓ in this tree and constructs the corresponding sets J_i^j for $i = 1, 2 \ldots g_j$. We also have the information sets I_i^j.

After this branch has been discovered we ask, now in the ℓ common decision tree, for the partner of any chosen element in $I^{j*} = \cup_{i=1}^{g_j} I_i^j$. The answers jointly with the matching pairs from π_2 in I^{j*} forms a new information set I^{j*+}. This information set is added to I^+ and we look for the next formula that needs a decision tree of depth at least ℓ. It might be the same formula F^{m_j}, but in such a case a later branch.

The sets J_i^j again have independent supports and there is no problem to construct ρ^* which is defined by ρ jointly with $\cup_{i,j} J_i^j$. Let $J^{j*} = \cup_i J_i^j$ be the information set obtained during the jth stage. It contains at least ℓ matched pairs but let us assume that the true number is s_j.

The external information determines which formulas where processed in the ℓ common decision tree. There are at most $M^{4s/\ell}$ possibilities and this gives an extra factor in the cost of the external information.

Once we know the value of m_j we can run the reconstruction process based on F^{m_j} exactly as in the single switching lemma and in particular reconstruct the sets I_i^j and J_i^j. The first element of each edge in J_i^j is identified at cost at most $O(t)$ and the second at cost at most $O(\Delta)$. To specify their partners inside I_i^j costs at most another factor $O(\log n)^2$. For each element in I_i^j we need to later specify a partner inside I^{*j+} and thus for the four mini-squares just found this gives an extra factor of $O(\log n)^4$. This implies that the total external cost at stage j is at most $2^{O(s_j)}(\log n)^{6s_j}(t\Delta)^{s_j}$.

As any edge in J_i^j might result in four queries in the common decision tree, if we have $4s$ questions in the this tree, then $\sum s_j \geq s$ and let us for convenience assume we have equality. The total cost of external information is in such a case $2^{O(s)}(\log n)^{6s}(t\Delta)^s$. By the same argument as in the single switching lemma case, the ratio of the number of matching with $k-s$ elements and the number with k elements is $2^{O(s)}(\log n/\Delta^2)^s$.

We conclude that the fraction of quadruples τ, U π_2, B that can reconstructed for a single choice of which formulas to process is bounded by

$$\Delta 2^{O(s)}((\log n)^7 t\Delta^{-1})^s.$$

This combined the the above claimed bound $M^{4s/\ell}$ for the number of possible choices of $4s/\ell$ formulas to process completes the proof of the lemma. ∎

VII. The lower bound for number of lines

We finally arrive at our second main theorem.

Theorem 7.1: Assume that $d \leq O(\frac{\log n}{\log\log n})$ and let n be an odd integer and M a parameter. Then any depth-d Frege proof of the one-to-one PHP on the $n \times n$ grid where each line is of size M requires at least

$$\exp\left(\Omega\left(\frac{n}{(\log M)^{2d-1}(\log n)^{O(1)}}\right)\right)$$

lines.

Proof: The proof is analogous to the proof in the full version of [HR22] and let us only sketch it, mostly giving parameters. We have $t = 1$ in the first iteration but in later iterations we use $t = \ell = \Theta(\log M)$ and hence the first factor $M^{4s/\ell}$ turns in to $2^{O(s)}$ and can be included in the constant.

We have $\Delta = \Theta((\log n)^7 t)$ to ensure that the base of the exponent (including the contribution from the $M^{4s/\ell}$ factor) is bounded by $\frac{1}{4}$. In the ith iteration we use $s_i = 2^i \log N$ where N is the number of lines in the proof. In the ith iteration we need to apply the switching lemma $N2^{\sum_{j=0}^{i-1} s_j} = N^{2^i}$ times. In each of the lines and in each leaf of the common decision tree that we have created so far and which is of depth $\sum_{j=0}^{i-1} s_j$.

After d rounds of restrictions we have reduced the size of the PHP from n to $n/(\log n)^{O(d)}(\log M)^{2d-2}$. If this is larger than $10(\sum_{j=0}^{d} s_j + \log M)$ we can conclude that there is no proof and this gives the bound of the theorem. ∎

VIII. Final words

It seems that to be able to prove a switching lemma for a space of restrictions one essential property is that for any given variable, the probability of setting it to either constant should be significantly higher than keeping it undetermined. In the unrestricted PHP the probability that any variable is true is about $\frac{1}{n}$ and to make the probability of the variables remaining undetermined smaller we must go from size n to a size smaller than $\sqrt{n}$. With such a quick decrease in size we can only apply the switching lemma $O(\log\log n)$ times. In the graph PHP on the grid the probability of a variable being true to about $\frac{1}{4}$ and at the same time the probability of keeping a variable undetermined is about $\log n/\Delta$. It is harder to preserve a graph PHP, but this is made possible by using augmenting paths as the new variables. Thus it seems that both decision were forced upon us but certainly there might be other possibilities and it would be interesting to see alternative proofs for a switching lemma for a space of restrictions that preserve the PHP.

In this paper we have proved yet another switching lemma and it might not even be the last one. There are many remaining questions in both proof complexity and circuit complexity and some might be attackable by these types of techniques. It would be very interesting, however, to find a different technique to attack questions of Frege proofs where each formula is relatively simple.

While proofs of switching lemmas are non-trivial, the properties of the proof we use are rather limited. In the assumed proof of contradiction, after the restrictions, the proof, more or less, only contain formulas of constant size. It is not difficult to see that such a proof cannot find a contradiction for a set of axioms that are locally consistent. It would be interesting with a reasoning that used more interesting properties of being a proof.

Acknowledgments

I am deeply grateful to Svante Janson for suggesting the proof of Lemma 3.6. I am also very grateful to Per Austrin, Kilian Risse, Ben Rossman, and Aleksa Stanković for a sequence of discussions that lead to the start of the ideas for this paper. The research leading to this publication was supported by the Knut and Alice Wallenberg foundation.

References

[AAG01] N. Galesi A. Atserias and R. Gavaldà. Monotoe proofs of the pigeon hole principle. *Mathematical Logic Quaterly*, 47:461–474, 2001.

[Ajt94] Miklós Ajtai. The complexity of the pigeonhole principle. *Combinatorica*, 14(4):417–433, 1994. Preliminary version in *FOCS '88*.

[Bol86] B. Bollobás. *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability*. Cambridge University Press, 1986.

[BPU92] Stephen Bellantoni, Toniann Pitassi, and Alasdair Urquhart. Approximation and small-depth frege proofs. *SIAM J. Comput.*, 21:1161–1179, 1992.

[FSS84] M. Furst, J.B. Saxe, and M. Sipser. Parity, circuits and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[Hak85] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297 – 308, 1985.

[Hås86] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, pages 6–20, New York, NY, USA, 1986. ACM.

[Hås14] J. Håstad. On the correlation of parity and small-depth circuits. *SIAM Journal on Computing*, 43:1699–1708, 2014.

[Hås20] J. Håstad. On small-depth frege proofs for tseitin for grids. *Journal of the ACM*, 68:1–31, 2020.

[HR22] J. Håstad and K. Risse. On bounded depth proofs for tseitin formulas on the grid; revisited. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1138–1149, 2022. Full version is available at ArXiv:2209.05839 https://arxiv.org/abs/2209.05839.

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithms for AC^0. In *Proceeding of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 961–972, 2012.

[KPW95] Jan Krajíček, Pavel Pudlák, and Alan Woods. An exponential lower bound to the size of bounded depth frege proofs of the pigeonhole principle. *Random Structures & Algorithms*, 7(1):15–39, 1995.

[PBI93] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993. Preliminary version in *STOC '92*.

[PRST16] Toniann Pitassi, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. Poly-logarithmic frege depth lower bounds via an expander switching lemma. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 644–657, New York, NY, USA, 2016. Association for Computing Machinery.

[PRT21] T. Pitassi, P. Ramakrishnan, and L. Tan. Tradeoffs for small-depth frege proofs. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 445–456, Los Alamitos, CA, USA, feb 2021. IEEE Computer Society.

[Raz95] A. A. Razborov. *Bounded Arithmetic and Lower Bounds in Boolean Complexity*, pages 344–386. Birkhäuser Boston, Boston, MA, 1995. Editors Peter Clote and Jeffrey Remmel.

[RST15] Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1030–1048, 2015.

[Sip83] M. Sipser. Borel sets and circuit complexity. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 61–69, New York, NY, USA, 1983. ACM.

[Tse68] G. S. Tseitin. On the complexity of derivation in the propositional calculus. In A. O. Slisenko, editor, *Studies in constructive mathematics and mathematical logic, Part II*, 1968.

[UF96] Alasdair Urquhart and Xudong Fu. Simplified lower bounds for propositional proofs. *Notre Dame Journal of Formal Logic*, 37(4):523–544, 1996.

[Yao85] A. C-C. Yao. Separating the polynomial-time hierarchy by oracles. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 1 –10, oct. 1985.