On the Pigeonhole and Related Principles in Deep Inference and Monotone Systems

Anupam Das
INRIA & University of Bath
anupam.das@inria.fr

Abstract

We construct quasipolynomial-size proofs of the propositional pigeonhole principle in the deep inference system KS, addressing an open problem raised in previous works and matching the best known upper bound for the more general class of monotone proofs.

We make significant use of monotone formulae computing boolean threshold functions, an idea previously considered in works of Atserias et al. The main construction, monotone proofs witnessing the symmetry of such functions, involves an implementation of merge-sort in the design of proofs in order to tame the structural behaviour of atoms, and so the complexity of normalization. Proof transformations from previous work on atomic flows are then employed to yield appropriate KS proofs.

As further results we show that our constructions can be applied to provide quasipolynomial-size KS proofs of the parity principle and the generalized pigeonhole principle. These bounds are inherited for the class of monotone proofs, and we are further able to construct $n^{O(\log \log n)}$-size monotone proofs of the weak pigeonhole principle with $(1+\varepsilon)n$ pigeons and n holes for $\varepsilon = 1/\log^k n$, thereby also improving the best known bounds for monotone proofs.

Categories and Subject Descriptors F.4.1 [*Mathematical Logic*]: Proof Theory

Keywords Pigeonhole Principle, Deep Inference, Monotone Proofs, Atomic Flows

1. Introduction

The *pigeonhole principle* states that if m pigeons are sitting in n holes, and $m > n$, then two pigeons must be in the same hole. It can be expressed in propositional logic as follows,

$$\mathsf{PHP}_n^m : \quad \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} p_{ij} \rightarrow \bigvee_{j=1}^{n} \bigvee_{i=1}^{m-1} \bigvee_{i'=i+1}^{m} p_{ij} \wedge p_{i'j}$$

where p_{ij} should be interpreted as "pigeon i sits in hole j". [1] This encoding forms a class of propositional tautologies, for $m > n$, that has become a benchmark in proof complexity [25]. For the case of $m = n + 1$ many propositional proof systems, such as the cut-free sequent calculus, Resolution and bounded-depth Frege, only have proofs of size exponential in n [22] [24], whereas small proofs (of size polynomial in n) have been constructed for Frege systems, and so also sequent calculi with cut [11].

This paper presents a novel proof structure for PHP_n^m, inspired by previous works of Atserias et al. [2] [3], implemented in a representation of monotone proofs[2] as rewriting derivations [19]. Consequently, we obtain quasipolynomial[3]-size proofs of PHP_n^{n+1} in the minimal deep inference system for propositional logic, KS. This answers questions previously raised in [9] [19] [26] [13] on the complexity of KS proofs of PHP_n^m by matching the best known bound for the more general class of monotone proofs [2].

By making certain generalizations we are able to apply our methods to obtain quasipolynomial-size KS proofs of the parity principle and the generalized pigeonhole principle, bounds that are inherited by the class of monotone proofs. Finally we show that our proof structure can be applied to yield $n^{O(\log \log n)}$-size monotone proofs of $\mathsf{PHP}_n^{(1+\varepsilon)n}$ where $\varepsilon = 1/ = 1/\log^k n$ for $k > 1$, significantly improving the best known bound of $n^{O(\log n)}$ inherited from proofs of PHP_n^{n+1} in [2].[4] We point out that this is the first example where considerations in the complexity of deep inference have yielded improved results for more mainstream systems in proof complexity.

Deep inference systems for classical propositional logic were introduced by Guglielmi et al. [16] [8] and, despite significant progress in recent years on the complexity of deep inference, the classification of the system KS remains open.

In [9] it was shown that KS polynomially simulates (tree-like) cut-free sequent calculi but not vice-versa. This result was strengthened in [13] where it was shown that KS polynomially simulates certain fragments of Resolution and dag-like cut-free sequent calculi, and it was also shown that these systems, as well as bounded-depth Frege systems, cannot polynomially simulate KS. This work made significant use of proof transformations induced by certain graph rewriting techniques from [17]. In this way the complexity of normalizing a monotone proof to a KS proof was reduced to count-

[1] Notice that the above formula allows the mapping from pigeons to holes to be many-many. Additional restrictions can be placed on the mapping, demanding that it is a function or that it is onto, resulting in a logically weaker formula, but here we consider only the version above.

[2] A monotone proof is a proof in the sequent calculus free of negation-steps.

[3] A quasipolynomial in n is a function of size $n^{\log^{\Theta(1)} n}$.

[4] This result also supports the more general conjecture in the community that the class of monotone proofs polynomially simulates Frege systems [3] [20] [21].

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.
Copyright © 2014 ACM 978-1-4503-2886-9…$15.00.
http://dx.doi.org/10.1145/2603088.2603164

ing the number of paths in the associated *atomic flow*, the graph obtained by tracing the journey of each atom through the proof.

It was asked in [26] and [9] whether polynomial-size proofs of PHP_n^m exist in KS, and in [26] it was conjectured that no polynomial-size proofs exist. On the other hand, in [19] Jeřábek gives proofs in an extended system of weaker variants of the pigeonhole principle, where the mapping from pigeons to holes is required to be functional or onto, which normalize to KS proofs of polynomial size [13]. He uses an elegant black-box construction relying on the existence of the aforementioned Frege proofs, although he notes that this method does not seem to generalize to PHP_n^m.

In this work we rely heavily on a propositional encoding of *threshold functions*, yielding formulae that count how many of their arguments are true, and our construction is inspired by the monotone proofs of PHP_n^m given by Atserias et al. [2]. They use the same threshold formulae as us but our main construction, short proofs that permute the arguments of a threshold formula, is considerably more involved than the analogous construction in their paper due to technicalities of the weaker system KS. The tradeoff is that this more sophisticated proof structure enables us to later achieve the aforementioned improvement in upper bounds on the size of monotone proofs for the weak pigeonhole principle.

In [2] simple proofs are provided for each transposition, whence the result follows since each permutation can be expressed as a product of at most polynomially many transpositions, resulting in monotone proofs whose atomic flows have polynomial length. However due to this length bound such proofs normalize to exponential-size KS proofs under the aforementioned transformations. Instead we notice in Sect. 3 that the specific permutation required, corresponding to the transposition of a matrix, has a particularly simple decomposition into logarithmically many *interleavings*, which we implement as monotone proofs whose atomic flows have polylogarithmic length and hence normalize to KS proofs in quasipolynomial time.

In Sect. 4 we generalize this construction by noticing that any permutation can be decomposed into a product of logarithmically many *riffle shuffles*; this is equivalent to the action of applying merge-sort to the inverse of a permutation. In Sect. 5, we show that these techniques can be applied to yield the aforementioned proofs of the parity principle, generalized pigeonhole principle and the weak pigeonhole principle. In the final result the polylogarithmic length of the atomic flows of our proof structure is crucial since it allows us to use smaller monotone formulae that only *approximate* threshold functions, and to maintain a sufficiently accurate approximation throughout the various permutations of their arguments.

We omit certain proofs, particularly in Sects. 4 and 5, due to space restrictions. Full proofs can be found in an extended version of this paper [14].

2. Preliminaries

Deep inference systems for classical logic were introduced in [8] and studied in detail in [6] and [9]. The representation of proofs we use here was introduced in [18].

2.1 Propositional Logic

Propositional formulae are constructed freely from atoms (propositional variables and their duals), also known as literals, over the basis $\{\top, \bot, \wedge, \vee\}$, with their usual interpretations. The variables a, b, c, d range over atoms, with $\bar{a}, \bar{b}, \dots$ denoting their duals, and A, B, C, D range over formulae. There is no object-level symbol for negation; instead we may write $\bar{A}$ to denote the De Morgan dual of A, obtained by the following rules:

$$\bar{\bot} = \top, \quad \bar{\top} = \bot, \quad \bar{\bar{a}} = a, \quad \overline{A \vee B} = \bar{A} \wedge \bar{B}, \quad \overline{A \wedge B} = \bar{A} \vee \bar{B}$$

For convenience, we consider formulae equivalent under the smallest equivalence relation generated by the equations below.

$$\begin{array}{cc} [A \vee B] \vee C = A \vee [B \vee C] & A \vee \bot = A \\ (A \wedge B) \wedge C = A \wedge (B \wedge C) & A \wedge \top = A \\ A \vee B = B \vee A & \top \vee \top = \top \\ A \wedge B = B \wedge A & \bot \wedge \bot = \bot \end{array}$$

$$\text{If } A = B,\ \star \in \{\wedge, \vee\}, \text{ then } C \star A = C \star B$$

For this reason we generally omit internal brackets of a formula, under associativity, as well as external brackets. For clarity we also use square brackets $[,]$ for disjunctions and round ones $(,)$ for conjunctions.

Remark 1 (Equality)**.** Equality of formulae $=$, as defined above, is usually implemented as an inference rule in deep inference. It is decidable in polynomial time [9], and whether it is implemented as an inference rule or equivalence relation is purely a matter of convention. Nonetheless we sometimes use it as a 'fake' inference rule, to aid the reader.

It will sometimes be convenient to represent the arguments of a boolean function as a vector or matrix of atoms. However the order in which the atoms are to be read is sensitive, and so we introduce the following notation.

Notation 2 (Vectors and Matrices of Variables)**.** We use bold lowercase letters $\boldsymbol{a}, \boldsymbol{b}, \dots$ to denote (row-)vectors of atoms and bold uppercase letters $\boldsymbol{A}, \boldsymbol{B}, \dots$ to denote matrices of atoms. Vectors are read in their natural order, and we associate a matrix with the vector obtained by reading it rows-first. In this way the transpose of a matrix is equivalent to the vector obtained by reading it columns-first.

The notation $(\boldsymbol{a}, \boldsymbol{b})$ denotes the horizontal concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and compound matrices are similarly written in the usual way. The notation $(a_i)_{i=1}^n$ denotes the vector $(a_1, \dots, a_n)$.

Definition 3 (Rules and Systems)**.** An *inference rule* is a binary relation on formulae decidable in polynomial time, and a *system* is a set of rules. We define the deep inference system SKS as the set of all inference rules in Fig. 1, and also the subsystem $\mathsf{KS} = \{\mathsf{ai}\!\downarrow, \mathsf{aw}\!\downarrow, \mathsf{ac}\!\downarrow, \mathsf{s}, \mathsf{m}\}$. Note in particular the distinction between variables for atoms and formulae.

Remark 4. It is worth pointing out that the formulation of deep inference with units is very convenient for proof-theoretic manipulation of derivations, and we exploit this throughout. However we could equally formulate our systems without units with no significant change in complexity; this approach is taken in [26] and the equivalence of these two formulations is shown in [12].

Definition 5 (Proofs and Derivations)**.** We define derivations, and premiss and conclusion functions, pr, cn resp., inductively:

1. Each formula A is a derivation with premiss and conclusion A.
2. If Φ, Ψ are derivations and $\star \in \{\wedge, \vee\}$ then $\Phi \star \Psi$ is a derivation with premiss $\mathsf{pr}(\Phi) \star \mathsf{pr}(\Psi)$ and conclusion $\mathsf{cn}(\Phi) \star \mathsf{cn}(\Psi)$.
3. If Φ, Ψ are derivations and $\rho\,\dfrac{\mathsf{cn}(\Phi)}{\mathsf{pr}(\Psi)}$ is an instance of a rule ρ then $\rho\,\dfrac{\Phi}{\Psi}$ is a derivation with premiss $\mathsf{pr}(\Phi)$ and conclusion $\mathsf{cn}(\Psi)$.

If $\mathsf{pr}(\Phi) = \top$ then we call Φ a *proof*. If Φ is a derivation where all inference steps are instances of rules in a system $\mathcal{S}$ with premiss A, conclusion B, we write $\begin{array}{c} A \\ \Phi \Vert \mathcal{S} \\ B \end{array}$. Furthermore, if $A = \top$, i.e. Φ is a proof in a system $\mathcal{S}$, we write $\begin{array}{c} \Phi \Vert \mathcal{S} \\ B \end{array}$.

$$
\begin{array}{cccc}
\multicolumn{3}{c}{\text{Atomic structural rules}} & \text{Logical rules} \\[1ex]
\mathsf{ai}\!\downarrow \dfrac{\top}{a \vee \bar{a}} & \mathsf{aw}\!\downarrow \dfrac{\bot}{a} & \mathsf{ac}\!\downarrow \dfrac{a \vee a}{a} & \mathsf{s}\, \dfrac{A \wedge [B \vee C]}{(A \wedge B) \vee C} \\
\textit{identity} & \textit{weakening} & \textit{contraction} & \textit{switch} \\[2ex]
\mathsf{ai}\!\uparrow \dfrac{a \wedge \bar{a}}{\bot} & \mathsf{aw}\!\uparrow \dfrac{a}{\top} & \mathsf{ac}\!\uparrow \dfrac{a}{a \wedge a} & \mathsf{m}\, \dfrac{(A \wedge B) \vee (C \wedge D)}{[A \vee C] \wedge [B \vee D]} \\
\textit{cut} & \textit{coweakening} & \textit{cocontraction} & \textit{medial}
\end{array}
$$

Figure 1. Rules of the deep inference system SKS.

We extend our structural rules beyond atoms, to general formulae, below.

Proposition 6 (Generic Rules)**.** *Each rule below has polynomial-size derivations in the system containing* s, m, *and its respective atomic structural rule.*

$$
\begin{array}{ccc}
\mathsf{i}\!\downarrow \dfrac{\top}{A \vee \bar{A}} & \mathsf{w}\!\downarrow \dfrac{\bot}{A} & \mathsf{c}\!\downarrow \dfrac{A \vee A}{A} \\
\mathsf{i}\!\uparrow \dfrac{A \wedge \bar{A}}{\bot} & \mathsf{w}\!\uparrow \dfrac{A}{\top} & \mathsf{c}\!\uparrow \dfrac{A}{A \wedge A}
\end{array}
$$

Proof Sketch. See [8] for full proofs. We just consider the case for contraction, since that is the only structural rule Gentzen calculi cannot reduce to atomic form [5]. The proof is by induction on the depth of the conclusion of a c↓-step.

$$
\mathsf{c}\!\downarrow \frac{[A \vee B] \vee [A \vee B]}{A \vee B} \quad \to \quad = \frac{[A \vee B] \vee [A \vee B]}{\mathsf{c}\!\downarrow \dfrac{A \vee A}{A} \vee \mathsf{c}\!\downarrow \dfrac{B \vee B}{B}}
$$

$$
\mathsf{c}\!\downarrow \frac{(A \wedge B) \vee (A \wedge B)}{A \wedge B} \quad \to \quad \mathsf{m} \frac{(A \wedge B) \vee (A \wedge B)}{\mathsf{c}\!\downarrow \dfrac{A \vee A}{A} \wedge \mathsf{c}\!\downarrow \dfrac{B \vee B}{B}}
$$

Note that the case for c↑ is dual to this: one can just flip the derivations upside down and replace every formula with its De Morgan dual. c↓-steps become c↑-steps and s and m steps remain valid. □

We often use these 'generic' rules in proof constructions, which should be understood as abbreviations for the derivations mentioned above.

Definition 7 (Complexity)**.** The *size* of a derivation Φ, denoted $|\Phi|$, is the number of atom occurrences in it. For a vector $\boldsymbol{a}$ or matrix $\boldsymbol{A}$, let $|\boldsymbol{a}|$, $|\boldsymbol{A}|$ denote its number of elements, respectively.

We will generally omit complexity arguments when they are routine, for convenience. However we outline the main techniques used to control complexity in the following sections.

2.2 Monotone and Normal Derivations

We define monotone and normal derivations and relate them to proof systems in deep inference. We point out that the notion of monotone derivation given here is polynomially equivalent to the tree-like monotone sequent calculus [19], and so is consistent with the usual terminology from the point of view of proof complexity.

Definition 8. A derivation is *monotone* if it does not contain the rules ai↓, ai↑. A monotone derivation is said to be *normal* if it has the following shape:

$$
\begin{array}{c}
A \\
\| \{\mathsf{aw}\!\uparrow, \mathsf{ac}\!\uparrow, \mathsf{s}, \mathsf{m}\} \\
B \\
\| \{\mathsf{aw}\!\downarrow, \mathsf{ac}\!\downarrow, \mathsf{s}, \mathsf{m}\} \\
C
\end{array}
$$

The significance of normal derivations is that they can be efficiently transformed into KS-proofs of the implication they derive, as demonstrated in the following proposition.

Proposition 9. *A normal derivation* $\begin{array}{c} A \\ \Phi \| \{\mathsf{aw}\!\uparrow, \mathsf{ac}\!\uparrow, \mathsf{s}, \mathsf{m}\} \\ B \\ \Psi \| \{\mathsf{aw}\!\downarrow, \mathsf{ac}\!\downarrow, \mathsf{s}, \mathsf{m}\} \\ C \end{array}$ *can be transformed in linear time to a* KS*-proof of* $\bar{A} \vee C$.

Proof Sketch. Define the derivation $\begin{array}{c} \bar{B} \\ \bar{\Phi} \| \{\mathsf{aw}\!\downarrow, \mathsf{ac}\!\downarrow, \mathsf{s}, \mathsf{m}\} \\ \bar{A} \end{array}$ by flipping Φ upside-down, replacing every atom with its dual, ∧ for ∨ and vice-versa. aw↑-steps become aw↓-steps, ac↑-steps become ac↓-steps and s and m steps remain valid. Now construct the required derivation: $\mathsf{i}\!\downarrow \dfrac{\top}{\begin{array}{ccc} \bar{B} & & B \\ \bar{\Phi} \| & \vee & \Psi \| \\ \bar{A} & & C \end{array}}$. □

We emphasize that it is the existence of an 'intermediate' formula in normal derivations, e.g. B in the proof above, that allows us to isolate all the ↑ steps and flip them into ↓ steps, resulting in a KS proof. If we started with an arbitrary monotone derivation there may be no such formula, and so any choice of an intermediate formula would also flip some ↓ steps into ↑ steps.

2.3 Atomic Flows and Normalization

We are particularly interested in those monotone derivations that can be efficiently transformed to normal ones. A thorough analysis of the complexity of such transformations is carried out in [13] in the setting of graph rewriting. We state informally the main concepts and results here.

Atomic flows and various rewriting systems on them were introduced formally in [17].

Definition 10 (Atomic Flows and Dimensions)**.** The *atomic flow*, or simply flow, of a monotone derivation is the vertically directed graph obtained by tracing the paths of each atom through the derivation, designating the creation, destruction and duplication of

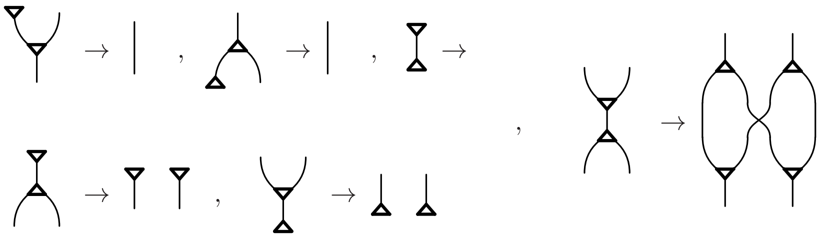
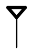
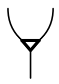
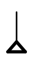
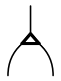


Figure 2. Graph rewriting rules for atomic flows.

atom occurrences by the following nodes:

$\mathsf{aw}\!\downarrow\dfrac{\bot}{a}\quad\rightarrow$  $\qquad\mathsf{ac}\!\downarrow\dfrac{a\vee a}{a}\quad\rightarrow$ 

$\mathsf{aw}\!\uparrow\dfrac{a}{\top}\quad\rightarrow$  $\qquad\mathsf{ac}\!\uparrow\dfrac{a}{a\wedge a}\quad\rightarrow$ 

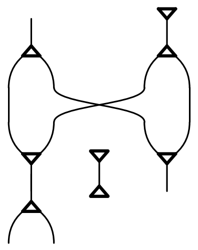
We do not have nodes for s and m steps since they do not create, destroy or duplicate any atom occurrences, and we generally consider flows equivalent up to continuous deformation preserving the vertical order of edges.

The *size* of a flow is its number of edges. The *length* of a flow is the maximum number of times the node type changes in a (vertically directed) path. The *width* of a flow is the maximum number of input or output edges in a subgraph of a connected component.

For intuition, the width of a flow can be thought of as a measure of how much a configuration of $\mathsf{ac}\!\uparrow$ nodes increases the number of edges in a connected component before a configuration of $\mathsf{ac}\!\downarrow$ nodes decreases it.

Example 11. We give an example of a monotone derivation and its flow below:

$$\mathsf{s}\,\dfrac{\mathsf{m}\,\dfrac{\mathsf{ac}\!\uparrow\dfrac{a}{a\wedge a}\vee\mathsf{ac}\!\uparrow\dfrac{\mathsf{aw}\!\uparrow\dfrac{\bot}{a}}{a\wedge a}}{=\dfrac{\mathsf{ac}\!\downarrow\dfrac{a\vee a}{a}}{a\vee\mathsf{aw}\!\downarrow\dfrac{\bot}{b}}\wedge\mathsf{ac}\!\downarrow\dfrac{a\vee a}{a}}}{\mathsf{ac}\!\uparrow\dfrac{a}{a\wedge a}\vee =\dfrac{\mathsf{aw}\!\uparrow\dfrac{b}{\top}\wedge a}{a}}$$



The flow has length 3, measured from the top-right $\mathsf{aw}\!\downarrow$ node to the bottom-left $\mathsf{ac}\!\uparrow$ node, and width 4, measured either as the outputs of the two top $\mathsf{ac}\!\uparrow$ nodes or the inputs of the two bottom $\mathsf{ac}\!\downarrow$ nodes.

Observation 12. *A normal derivation has flow length* 1.

Theorem 13 (Normalization)**.** *A monotone derivation Φ whose flow has width w and length l can be transformed into a normal derivation of size $|\Phi|\cdot w^{l+O(1)}$, preserving premiss and conclusion.*

While the proof of the above theorem can be found in [13], we outline the main ideas to give the reader an intuition of the argument.

Proof Sketch. The graph rewriting rules in Fig. 2 induce transformations on monotone derivations by consideration of the corresponding rule permutations; note that, due to atomicity of the structural rules, permutations with logical steps are trivial. The system is terminating and the flows of normal derivations are all normal forms of this system. Each rewrite step preserves the number of maximal paths between pending edges, and a normal derivation has size polynomial in this measure.

Consequently the complexity of normalizing a monotone derivation is polynomial in its size and the number of maximal paths in its flow, and this is estimated by the given bound. □

Notice, in particular, that any rewrite derivation on atomic flows acts independently on different connected components. Therefore the complexity of normalization is determined by the structural behaviour of individual atoms - there is no interaction between distinct atoms during normalization.

Finally, most of the proofs in this work are inductions, and for the base cases it will typically suffice to build any monotone proof of a single formula or simple class of formulae, since we are interested in how the size (or width, length) of the proofs grow and not their initial values. For this reason, the following result will be useful, and we implicitly assume it when omitting base cases of inductions.

Proposition 14 (Monotone Implicational Completeness)**.** *Let A, B be negation-free formulae such that $A \rightarrow B$ is valid. Then there is a monotone derivation* $\begin{array}{c}A\\ \|\\ B\end{array}$.

Proof Sketch. Construct a disjunctive normal form A' of A and conjunctive normal form B' of B by distributivity. Note that all distributivity laws are derivable by Dfn. 19 and duality, so there are monotone derivations $\begin{array}{c}B'\\ \|\\ B\end{array}$ and $\begin{array}{c}A\\ \|\\ A'\end{array}$. Clearly each conjunction of A' logically implies each disjunction of B' and so there must be derivations in $\{\mathsf{aw}\!\downarrow, \mathsf{aw}\!\uparrow\}$ witnessing this fact. Using these derivations and applying $\mathsf{c}\!\downarrow, \mathsf{c}\!\uparrow$ appropriately we can construct a monotone derivation $\begin{array}{c}A'\\ \|\\ B'\end{array}$, whence the result follows by sequential composition of these derivations. □

By appealing to Thm. 13 we then obtain the following result.

Corollary 15. *Normal derivations are monotone implicationally complete.*

3. Short Proofs of the Pigeonhole Principle

Throughout this section the variables m and n are powers of 2 and $m \leq n$. All proofs in this section are monotone unless otherwise mentioned.

3.1 Threshold Formulae and Permutations

Threshold functions are a class of boolean functions $\mathsf{TH}_k^n : \{0,1\}^n \to \{0,1\}$ by $\mathsf{TH}_k^n(\sigma_1 \cdots \sigma_n) = 1$ just if $\sum_{i=1}^n \sigma_i \geq k$.

In this section we define quasipolynomial-size monotone formulae computing such functions and construct derivations whose flows have length $\log^{O(1)} n$ and width $O(n)$ that conduct certain permutations on the arguments of such formulae.

Definition 16 (Threshold Formulae)**.** We define the formulae,

$$\mathsf{th}_k^1(a) \quad := \begin{cases} \top & k = 0 \\ a & k = 1 \\ \bot & k > 1 \end{cases}$$

$$\mathsf{th}_k^{2n}(\boldsymbol{a}, \boldsymbol{b}) \quad := \bigvee_{i+j=k} \mathsf{th}_i^n(\boldsymbol{a}) \wedge \mathsf{th}_j^n(\boldsymbol{b})$$

for vectors $\boldsymbol{a}, \boldsymbol{b}$ of length n.

Observation 17. *th_k^n computes the threshold function TH_k^n, and has size $n^{O(\log n)}$ and depth $O(\log n)$.*

Definition 18 (Interleaving)**.** For $\boldsymbol{a} = (a_1, \ldots, a_n), \boldsymbol{b} = (b_1, \ldots, b_n)$ let $\boldsymbol{a} \mathbin{|||} \boldsymbol{b}$ denote the *interleaving* of $\boldsymbol{a}$ with $\boldsymbol{b}$: $(a_1, b_1, \ldots, a_n, b_n)$.

More generally, we denote by $\boldsymbol{a} \mathbin{|||}_m \boldsymbol{b}$ the m-interleaving:

$$(a_1, \ldots, a_m, b_1, \ldots b_m, \cdots, a_{n-m+1}, \ldots, a_n, b_{n-m+1}, \ldots, b_n)$$

Definition 19 (Distributivity)**.** We define distributivity rules as abbreviations for the following derivations:

$$\mathsf{dist}\uparrow: \qquad \mathsf{2{\cdot}s}\,\frac{\mathsf{c}\!\uparrow\dfrac{A}{A \wedge A} \wedge [B \vee C]}{(A \wedge B) \vee (A \wedge C)}$$

$$\mathsf{dist}\downarrow: \qquad \mathsf{m}\,\frac{(A \wedge B) \vee (A \wedge C)}{\mathsf{c}\!\downarrow\dfrac{A \vee A}{A} \wedge [B \vee C]}$$

Lemma 20. *There are monotone derivations,*

$$\begin{array}{c} \mathsf{th}_k^{2n}(\boldsymbol{a}, \boldsymbol{b}) \\ \Vert \\ \mathsf{th}_k^{2n}(\boldsymbol{a} \mathbin{|||}_m \boldsymbol{b}) \end{array}$$

whose flows have length $O(\log n)$ and width $O(n)$.

Proof. We use the following identity:

$$(\boldsymbol{a}, \boldsymbol{b}) \mathbin{|||}_m (\boldsymbol{c}, \boldsymbol{d}) = (\boldsymbol{a} \mathbin{|||}_m \boldsymbol{c}, \boldsymbol{b} \mathbin{|||}_m \boldsymbol{d})$$

We give an inductive step from n to $2n$ in Fig. 3 where derivations marked IH are obtained by the inductive hypothesis.

The $\mathsf{dist}\uparrow$ steps duplicate each atom at most r times and so, analyzing the associated flow, each inductive step adds $O(r)$ configurations of $\mathsf{ac}\!\uparrow$ and $\mathsf{ac}\!\downarrow$ nodes of width $O(r)$ on top of $O(r)$ copies of the inductive hypothesis in parallel. The induction terminates in $\log \frac{n}{m}$ steps, whence the bound on length is obtained. ☐

Observation 21. *For matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ of equal dimensions we have:*[5]

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} \boldsymbol{A}^{\mathsf{T}} & \boldsymbol{C}^{\mathsf{T}} \\ \boldsymbol{B}^{\mathsf{T}} & \boldsymbol{D}^{\mathsf{T}} \end{pmatrix}$$

Recall that a matrix of atoms is equivalent to the vector obtained from a rows-first reading of it.

[5] Of course, in any such situation, $\boldsymbol{A}$ and $\boldsymbol{D}$ will also have equal dimensions.

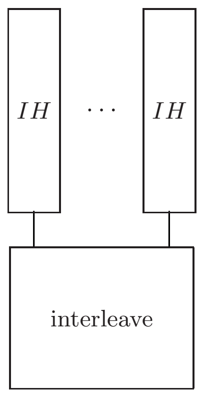
Theorem 22 (Transposition)**.** *There are monotone derivations,*

$$\begin{array}{c} \mathsf{th}_k^n(\boldsymbol{X}) \\ \Vert \\ \mathsf{th}_k^n(\boldsymbol{X}^{\mathsf{T}}) \end{array}$$

whose flows have length $O(\log^2 n)$ and width $O(n)$.

Proof. Let $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ be the four quadrants of $\boldsymbol{X}$. We give an inductive step from n to $2n$,

$$\begin{array}{c} \dfrac{\dfrac{\mathsf{th}_k^{2n}\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}}{\displaystyle\bigvee_{i+j=k} \left(\begin{array}{c} \mathsf{th}_i^n\left(\boldsymbol{A} \quad \boldsymbol{B}\right) \\ {\scriptstyle IH}\Vert \\ \mathsf{th}_i^n\begin{pmatrix} \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{B}^{\mathsf{T}} \end{pmatrix} \end{array} \wedge \begin{array}{c} \mathsf{th}_j^n\left(\boldsymbol{C} \quad \boldsymbol{D}\right) \\ {\scriptstyle IH}\Vert \\ \mathsf{th}_j^n\begin{pmatrix} \boldsymbol{C}^{\mathsf{T}} \\ \boldsymbol{D}^{\mathsf{T}} \end{pmatrix} \end{array} \right)}}{\mathsf{th}_k^{2n}\left(\begin{pmatrix} \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{B}^{\mathsf{T}} \end{pmatrix}, \begin{pmatrix} \boldsymbol{C}^{\mathsf{T}} \\ \boldsymbol{D}^{\mathsf{T}} \end{pmatrix}\right)} \\ {\scriptstyle \text{interleave}}\Vert \\ \mathsf{th}_k^{2n}\begin{pmatrix} \boldsymbol{A}^{\mathsf{T}} & \boldsymbol{C}^{\mathsf{T}} \\ \boldsymbol{B}^{\mathsf{T}} & \boldsymbol{D}^{\mathsf{T}} \end{pmatrix} \end{array}$$



where the derivations marked IH are obtained by the inductive hypothesis and Obs. 21, and the derivation marked 'interleave' is obtained by applying Lemma 20 to interleave the rows of the two matrices.

Analyzing the associated flow, each inductive step adds an interleaving below $O(k)$ copies of the inductive hypothesis in parallel, thereby adding $O(\log n)$ to the length and maintaining a width of $O(n)$, by Lemma 20. The induction terminates in $O(\log n)$ steps, whence the upper bound on length is obtained. ☐

3.2 From Threshold Formulae to the Pigeonhole Principle

The previous section showed that there are 'short' derivations transposing a matrix of arguments of a threshold formula. We show here how such derivations are used to obtain quasipolynomial-size proofs of the pigeonhole principle.

In this section almost all derivations are normal, so we omit their flows and complexity analysis.

Definition 23 (Pigeonhole Principle)**.** We define the following:

$$\mathsf{LPHP}_n := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n-1} p_{ij} \quad \mathsf{RPHP}_n := \bigvee_{j=1}^{n-1} \bigvee_{i=1}^{n} \bigvee_{i'=i+1}^{n} (p_{i'j} \wedge p_{ij})$$

$$\mathsf{PHP}_n := \mathsf{LPHP}_n \to \mathsf{RPHP}_n$$

Definition 24. Let $\bot_{mn}$ be the $(m \times n)$ matrix with the constant $\bot$ at every entry. Define $\boldsymbol{P}_n = \begin{pmatrix} (p_{ij}) & \bot_{n1} \end{pmatrix}$, with i, j ranging as in Dfn. 23. I.e. $\boldsymbol{P}_n$ is obtained by extending (p_{ij}) with an extra column of $\bot$-entries, so that it is a square matrix.

Our aim in this section is to prove the following theorem, from which we can extract proofs of PHP_n in KS by the results in earlier sections.

Theorem 25. *There are normal derivations,*

$$\begin{array}{c} \mathsf{LPHP}_n \\ \Vert \\ \mathsf{th}_n^{n^2}(\boldsymbol{P}_n) \end{array} \quad , \quad \begin{array}{c} \mathsf{th}_n^{n^2}(\boldsymbol{P}_n^{\mathsf{T}}) \\ \Vert \\ \mathsf{RPHP}_n \end{array}$$

of size $n^{O(\log n)}$.

$$
=\frac{\mathsf{th}^{4n}_r(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d})}{\displaystyle\bigvee_{s+t=r}\left(\begin{array}{c}
=\dfrac{\mathsf{th}^{2n}_s(\boldsymbol{a},\boldsymbol{b})}{\displaystyle\bigvee_{i+j=s}\mathsf{th}^n_i(\boldsymbol{a})\wedge\mathsf{th}^n_j(\boldsymbol{b})}\wedge=\dfrac{\mathsf{th}^{2n}_t(\boldsymbol{c},\boldsymbol{d})}{\displaystyle\bigvee_{k+l=t}\mathsf{th}^n_k(\boldsymbol{c})\wedge\mathsf{th}^n_l(\boldsymbol{d})}\\
\Big\Vert\,\mathsf{dist}\!\uparrow\\
\displaystyle\bigvee_{\substack{i+j=s\\k+l=t}}\left(\mathsf{th}^n_i(\boldsymbol{a})\wedge\mathsf{th}^n_j(\boldsymbol{b})\right)\wedge\left(\mathsf{th}^n_k(\boldsymbol{c})\wedge\mathsf{th}^n_l(\boldsymbol{d})\right)
\end{array}\right)}
$$
$$
=\frac{\displaystyle\bigvee_{s+t=r}\left(\begin{array}{c}
\displaystyle\bigvee_{\substack{i+k=s\\j+l=t}}\left(\mathsf{th}^n_i(\boldsymbol{a})\wedge\mathsf{th}^n_k(\boldsymbol{c})\right)\wedge\left(\mathsf{th}^n_j(\boldsymbol{b})\wedge\mathsf{th}^n_l(\boldsymbol{d})\right)\\
\Big\Vert\,\mathsf{dist}\!\downarrow\\
\begin{array}{ccc}
\mathsf{th}^{2n}_s(\boldsymbol{a},\boldsymbol{c}) & & \mathsf{th}^{2n}_t(\boldsymbol{b},\boldsymbol{d})\\
{\scriptstyle IH}\Big\Vert & \wedge & {\scriptstyle IH}\Big\Vert\\
\mathsf{th}^{2n}_s(\boldsymbol{a}\,|||_m\,\boldsymbol{c}) & & \mathsf{th}^{2n}_t(\boldsymbol{b}\,|||_m\,\boldsymbol{d})
\end{array}
\end{array}\right)}{\mathsf{th}^{4n}_r((\boldsymbol{a},\boldsymbol{b})\,|||_m\,(\boldsymbol{c},\boldsymbol{d}))}
$$

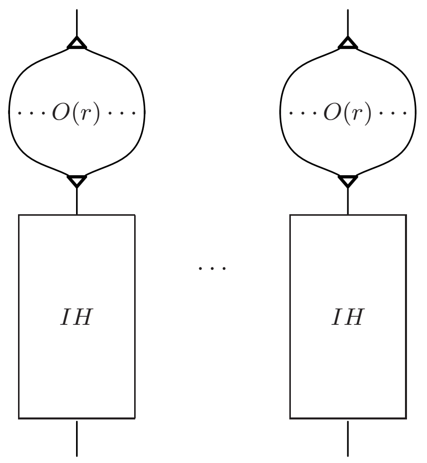


Figure 3. Interleaving the arguments of a threshold formula.

Before we can give a proof, we need some intermediate results. It should be pointed out that similar results were provided in [2] for the monotone sequent calculus, which could be translated to deep inference by [7] [19], but we include them for completeness. Indeed, similar results appeared in [10]. These intermediate results are fairly routine, and there is nothing intricate from the point of view of complexity.

Proposition 26. *For $l \geq k$ there are normal derivations,*

$$
\begin{array}{c}
\mathsf{th}^n_l(\boldsymbol{a})\\
\Big\Vert\\
\mathsf{th}^n_k(\boldsymbol{a})
\end{array}
$$

of size $n^{O(\log n)}$.

Proof. We give an inductive step from n to $2n$,

$$
=\frac{\mathsf{th}^{2n}_l(\boldsymbol{a},\boldsymbol{b})}{\begin{array}{c}
\displaystyle\bigvee_{i+j=l}\left(\begin{array}{ccc}
\mathsf{th}^n_i(\boldsymbol{a}) & & \mathsf{th}^n_j(\boldsymbol{b})\\
{\scriptstyle IH}\Big\Vert & \wedge & {\scriptstyle IH}\Big\Vert\\
\mathsf{th}^n_{i'}(\boldsymbol{a}) & & \mathsf{th}^n_{j'}(\boldsymbol{b})
\end{array}\right)\\
\Big\Vert\,\{\mathsf{w}\!\downarrow,\mathsf{c}\!\downarrow\}\\
\mathsf{th}^{2n}_k(\boldsymbol{a},\boldsymbol{b})
\end{array}}
$$

where i' and j' are chosen such that $i' \leq i$, $j' \leq j$ and $i' + j' = k$, and derivations marked IH are obtained by the inductive hypothesis. ☐

Lemma 27 (Evaluation)**.** *There are normal derivations,*

$$
\begin{array}{c}
\mathsf{th}^{2n}_{r+s}(\boldsymbol{a},\boldsymbol{b})\\
\Big\Vert\\
\mathsf{th}^n_{r+1}(\boldsymbol{a})\vee\mathsf{th}^n_s(\boldsymbol{b})
\end{array}
$$

of size $n^{O(\log n)}$.

Proof. Notice that if $i + j = r + s$ then $i > r$ or $j \geq s$. We give a construction in Fig. 4, where Φ and Ψ denote derivations obtained by Prop. 26. ☐

Lemma 28. *For vectors $\boldsymbol{a}^1, \dots, \boldsymbol{a}^m$ of atoms there are normal derivations,*

$$
\begin{array}{c}
\displaystyle\bigvee_{r=1}^{m}\mathsf{th}^n_k(\boldsymbol{a}^r)\\
\Vert\\
\mathsf{th}^{mn}_k(\boldsymbol{a}^r)^n_{r=1}
\end{array}
\quad,\quad
\begin{array}{c}
\displaystyle\bigwedge_{r=1}^{m}\mathsf{th}^n_k(\boldsymbol{a}^r)\\
\Vert\\
\mathsf{th}^{mn}_{mk}(\boldsymbol{a}^r)^n_{r=1}
\end{array}
$$

of size $n^{O(\log n)}$.

Proof. By induction on m. Simply apply $=$ and $\mathsf{w}\!\downarrow$ to fill out the formula. ☐

We are now in a position to prove Thm. 25.

Proof Sketch of Thm. 25. Repeatedly apply Lemma 27 to $\mathsf{th}^{n^2}_n(\boldsymbol{P}^{\intercal}_n)$, always setting $r = s$ or $r = s + 1$, until a disjunction of threshold formulae with n arguments each is obtained. By the ordering of the atoms in $\boldsymbol{P}^{\intercal}_n$ these threshold formulae will have as arguments $(p_{ij})^n_{i=1}$ for some j, or all $\bot$; in the latter case any such formula is equivalent to $\bot$, since the threshold will be at least 1.

In the former case, by the choice of r and s at each stage, we have that the threshold of each of these formulae is at least 2. From here Prop. 26 can be applied so that all thresholds are 2, whence RPHP_n can be easily derived.

For the other direction, construe each variable p_{ij} as a threshold formula $\mathsf{th}^1_1(p_{ij})$ and apply Lemma 28 to obtain a derivation from LPHP_n to $\mathsf{th}^{n^2}_n(\boldsymbol{P})$.

In both cases normality is established by the normalization procedure of Thm. 13. We have chained together finitely many normal derivations so the length of the associated flows is bounded by a constant, whence the upper bound on size is obtained. ☐

Theorem 29. *There are normal derivations,*

$$
\begin{array}{c}
\mathsf{LPHP}_n\\
\Vert\\
\mathsf{RPHP}_n
\end{array}
$$

of size $n^{O(\log^2 n)}$.

Proof. By Thms. 22 and 25 there are monotone derivations with same premiss and conclusion of length $O(\log^2 n)$ and width $O(n)$. The result then follows by Thm. 13. ☐

$$
= \frac{= \dfrac{\mathsf{th}^{2n}_{r+s}(\boldsymbol{a},\boldsymbol{b})}{\bigvee\limits_{i+j=r+s} \mathsf{th}^n_i(\boldsymbol{a}) \wedge \mathsf{th}^n_j(\boldsymbol{b})}}{\mathsf{s{\cdot}c}{\downarrow}\,\dfrac{\bigvee\limits_{\substack{i>r\\ j=r+s-i}} \left(\begin{array}{c} = \dfrac{\mathsf{th}^n_i(\boldsymbol{a}) \wedge \mathsf{w}{\uparrow}\dfrac{\mathsf{th}^n_j(\boldsymbol{b})}{\top}}{\mathsf{th}^n_i(\boldsymbol{a})} \\ \Phi \Big\Vert \\ \mathsf{th}^n_{r+1}(\boldsymbol{a}) \end{array} \right)}{\mathsf{th}^n_{r+1}(\boldsymbol{a})} \vee \mathsf{r{\cdot}c}{\downarrow}\,\dfrac{\bigvee\limits_{\substack{j\geq s\\ i=r+s-j}} \left(\begin{array}{c} = \dfrac{\mathsf{w}{\uparrow}\dfrac{\mathsf{th}^n_i(\boldsymbol{a})}{\top} \wedge \mathsf{th}^n_j(\boldsymbol{b})}{\mathsf{th}^n_j(\boldsymbol{b})} \\ \Psi \Big\Vert \\ \mathsf{th}^n_s(\boldsymbol{b}) \end{array} \right)}{\mathsf{th}^n_s(\boldsymbol{b})}}
$$

Figure 4. Proof of Lemma 27.

Corollary 30. *There are* KS *proofs of* PHP_n *of size* $n^{O(\log^2 n)}$.

Proof. By Prop. 9. $\square$

3.3 The Case when n is not a Power of 2

Though we have assumed that n is a power of 2 throughout this section, the proof is actually sufficient for all n, as pointed out in [2].

Definition 31. For $r \leq s$ given, define $\mathsf{LPHP}_s(r)$ by substituting $\perp$ for every atom p_{ij} where $i > r$ or $j \geq r$. Define $\mathsf{RPHP}_s(r)$ analogously.

Observation 32. *For all* $r \leq s$ *we have that* $\mathsf{LPHP}_s(r) = \mathsf{LPHP}_r$ *and* $\mathsf{RPHP}_s(r) = \mathsf{RPHP}_r$.[6] *Consequently a proof of* PHP_r *is just a proof of* PHP_n*, where* n *is the power of* 2 *such that* $r \leq n < 2r$.

4. Arbitrary Permutations

Interleavings by themselves do not form a generating set for the symmetric group, and so cannot be used to generate derivations for arbitrary permutations of arguments of threshold formulae. However a generalization of them, corresponding to the set of riffle shuffles on a deck of cards, do form such a set. In this section we show how they may be used to generate arbitrary permutations on the arguments of threshold formulae.

The proofs in this section are similar to those in Sect. 3, and so we omit them for brevity, instead providing the general proof structure as intermediate results.

Recall that our original definition of threshold formulae used a symmetric divide-and-conquer strategy, generated from a complete binary tree in the natural way. In this section it will be useful to have a more general definition of threshold formulae, based on any tree decomposition of the divide-and-conquer strategy.

Throughout this section we assume all trees are binary.

Definition 33. For a tree T, let $d(T)$ denote its depth, $l(T)$ its number of leaves and $|T|$ denote its number of nodes. For a binary tree T, let T_0 denote its left subtree (from the root) and T_1 its right. Thus any string $\sigma \in \{0,1\}^k$ determines a unique subtree T_σ of T, for $k \leq d(T)$.

Definition 34 (General Threshold Formulae)**.** For a binary tree T and vectors $\boldsymbol{a}, \boldsymbol{b}$ with $|\boldsymbol{a}| = l(T_0)$, $|\boldsymbol{b}| = l(T_1)$, define

$$\mathsf{th}^T_k(\boldsymbol{a},\boldsymbol{b}) = \bigvee_{i+j=k} \mathsf{th}^{T_0}_i(\boldsymbol{a}) \wedge \mathsf{th}^{T_1}_j(\boldsymbol{b})$$

with the base case the same as in Dfn. 16.

The following proposition gives an estimate of the size of these threshold formulae.

[6] Recall that formulae are equivalent up to $=$.

Proposition 35. *For a binary tree* T, $|\mathsf{th}^T_k(\boldsymbol{a})| = l(T)^{O(d(T))}$.

Proof. In the worst case, every level of the binary tree is full, whence the bound is obtained by Obs. 17. $\square$

What we define as a shuffle below corresponds to the common riffle method of shuffling a deck of cards: cut the deck anywhere, partitioning it into a left and right part, and then interleave these in any way, maintaining the relative order of cards in either partition. Under this analogy each card of the deck will correspond to a leaf of the tree determining a threshold formula.

Definition 36 (Cuts and Shuffles)**.** A *cut* of a vector $(a_1, \dots, a_n)$ is a pair $\{(a_1, \dots, a_m), (a_{m+1}, \dots, a_n)\}$. A *riffle shuffle*, or simply *shuffle*, of length n is a string $\sigma \in \{0,1\}^n$.

For a vector $\boldsymbol{a}$ and shuffle σ of length $|\boldsymbol{a}|$ we write $\sigma(\boldsymbol{a})$ to denote the natural action of σ on $\boldsymbol{a}$.

In the above definition, one should think of the 0s and 1s indicating whether a card is dropped from the left or right partition of the deck, with the cut determined by the number of 0s (or equivalently 1s).

Lemma 37 (Cutting)**.** *For any tree* T *and cut* $\{\boldsymbol{a}, \boldsymbol{b}\}$ *there are trees* S_0, S_1 *with* $d(S_0), d(S_1) \leq d(T)$ *such that there are monotone derivations,*

$$\begin{array}{c} \mathsf{th}^T_k(\boldsymbol{a},\boldsymbol{b}) \\ \Big\Vert \\ \bigvee\limits_{i+j=k} \mathsf{th}^{S_0}_i(\boldsymbol{a}) \wedge \mathsf{th}^{S_1}_j(\boldsymbol{b}) \end{array}$$

whose flows have length $O(d(T))$ *and width* $O(l(T))$.

Lemma 38 (Shuffling)**.** *Let* S *be a tree and* σ *a shuffle of length* $l(S)$*. There is a tree* T *with* $d(T) = O(d(S))$ *and monotone derivations,*

$$\begin{array}{c} \mathsf{th}^S_k(\boldsymbol{v}) \\ \Big\Vert \\ \mathsf{th}^T_k(\sigma(\boldsymbol{v})) \end{array}$$

whose flows have length $O(d(S)^2)$ *and width* $O(l(S))$.

Theorem 39 (Merge Sort)**.** *For any tree* S *and permutation* π *on* $\{1, \dots, l(S)\}$ *there is a tree* T *with* $d(T) = O(d(S))$ *and monotone derivations,*

$$\begin{array}{c} \mathsf{th}^S_k(a_{i\pi})^n_{i=1} \\ \Big\Vert \\ \mathsf{th}^T_k(a_i)^n_{i=1} \end{array}$$

whose flows have length $O(d(S)^3)$ *and width* $O(l(S))$.

Proposition 40 (Repartitionings)**.** *For trees S, T with the same number of leaves there are monotone derivations,*

$$\begin{array}{c}\mathsf{th}_k^S(\boldsymbol{a})\\ \| \\ \mathsf{th}_k^T(\boldsymbol{a})\end{array}$$

whose flows have length $O(d(S)^2)$ and width $O(l(S))$.

Corollary 41. *For any tree T and permutation π on $\{1,\ldots,l(T)\}$ there are normal derivations,*

$$\begin{array}{c}\mathsf{th}_k^T(a_i)_{i=1}^n\\ \| \\ \mathsf{th}_k^T(a_{i\pi})_{i=1}^n\end{array}$$

of size $l(T)^{O(d(T)^3)}$.

Proof. By Thm. 39, Prop. 40 and Thm. 13. ☐

5. Further Results and Applications

We give some examples of how the techniques developed in previous sections can be applied to yield further results, namely quasipolynomial-size normal proofs of the Generalized Pigeonhole principle and the Parity principle. Both bounds are also inherited for monotone proofs although, while these have not appeared in the literature, we point out that such monotone proofs could also have been constructed using the permutation arguments of Atserias et al. in [2].

More interestingly we provide $n^{O(\log\log n)}$-size monotone proofs for the *weak* pigeonhole principle, with $(1+\varepsilon)n$ pigeons and n holes for every $\varepsilon = 1/\log^{\Omega(1)} n$, improving the previous best known bound of $n^{O(\log n)}$ inherited from the proofs of PHP_n^{n+1} given in [2].

5.1 Generalized Pigeonhole Principle

If there are 45 hats that are either red or green, then there must be 23 of the same colour. This exemplifies a generalization of the pigeonhole principle where sufficiently many pigeons may guarantee more than two in some hole [15]. If $k+1$ pigeons in some hole are required then $nk+1$ pigeons are necessary, so this principle can be encoded as follows:

$$\bigwedge_{i=0}^{nk+1}\bigvee_{j=1}^{n} a_{ij} \rightarrow \bigvee_{i_r<i_{r+1}}\bigwedge_{r=1}^{k+1} a_{i_r j}$$

This formula has size $O(n^{k+1})$, polynomial for fixed k. If, however k is large relative to n, e.g. $n/2$ or $\sqrt{n}$, then one can always express the right hand side using threshold formulae to obtain an encoding of quasipolynomial-size.

It is simple to see that our proofs of PHP_n can be generalized to this class of tautologies, by the same arguments as in Sect. 3.

5.2 Parity Principle

The *parity principle* states that one cannot partition an odd-size set into pairs, and is usually encoded by the following tautologies,

$$\mathsf{PAR}_n \quad : \quad \bigwedge_{i=0}^{2n}\bigvee_{j\neq i} a_{\{i,j\}} \rightarrow \bigvee_{j\neq i>i'\neq j} a_{\{i,j\}} \wedge a_{\{i',j\}}$$

where $a_{\{i,j\}}$ should be interpreted as "element i is paired with element j".

These tautologies have similar structure to PHP_n, but in many proof systems these tautologies are in fact *harder* to prove. For example, in bounded-depth Frege systems PHP_n can be efficiently derived from PAR_n but not vice-versa [1] [4].

However, in KS, we can construct quasipolynomial-size proofs of PAR_n using similar methods to those for PHP_n, and we give an outline of these constructions in this subsection.

We omit proofs corresponding to basic properties of threshold functions, since they are fairly routine inductions of which Sect. 3 has given many examples, and also often do not specify precise orderings of variables or tree-structures of a threshold formulae, since these can all be reduced to any other in quasipolynomial time, by the results of Sect. 4.

Let LPAR_n and RPAR_n denote the left and right hand sides of PAR_n respectively. By a similar argument to Prop. 25 we obtain normal derivations of the following form,

$$\begin{array}{c}\mathsf{LPAR}_n\\ \| \\ \mathsf{th}_{2n+1}^{2n(2n+1)}(\boldsymbol{a}^2)\end{array}$$

where $\boldsymbol{a}^2$ is an appropriate sequence of the variables $a_{\{i,j\}}$ in which each variable occurs exactly twice, as in LPAR_n.

Let $(\boldsymbol{a},\boldsymbol{a})$ be a permutation of $\boldsymbol{a}^2$ so that each variable occurs exactly once in $\boldsymbol{a}$. Now we can construct the following derivation,

$$\begin{array}{c}\mathsf{th}_{2n+1}^{2n(2n+1)}(\boldsymbol{a}^2)\\ \text{permute}\,\| \\ \mathsf{th}_{2n+1}^{2n(2n+1)}(\boldsymbol{a},\boldsymbol{a})\\ \text{evaluate}\,\| \\ \mathsf{c}\!\downarrow \dfrac{\mathsf{th}_{n+1}^{n(2n+1)}(\boldsymbol{a}) \vee \mathsf{th}_{n+1}^{n(2n+1)}(\boldsymbol{a})}{\mathsf{th}_{n+1}^{n(2n+1)}(\boldsymbol{a})}\end{array}$$

where the derivation marked 'permute' applies the results of Sect. 4, namely Cor. 41, to permute the arguments of a threshold formula, and the derivation marked 'evaluate' is obtained by Lemma 27, setting $r = n$ and $s = n+1$.

Now notice that, if $n+1$ of the variables $a_{\{i,j\}}$ are true, i.e. we have $n+1$ pairs out of $2n+1$ variables, we must have some j which is paired with two distinct variables, and this can be realized as derivations,

$$\begin{array}{c}\mathsf{th}_{n+1}^{n(2n+1)}(\boldsymbol{a})\\ \| \\ \mathsf{RPAR}_n\end{array}$$

in a similar way to Prop. 25.

Chaining all these normal derivations together gives us monotone derivations $\begin{array}{c}\mathsf{LPAR}_n\\ \| \\ \mathsf{RPAR}_n\end{array}$ of quasipolynomial size and with flows of bounded length, and from here we can construct quasipolynomial-size KS-proofs of PAR_n in the usual way.

5.3 Monotone Proofs of the Weak Pigeonhole Principle

The results of this section provide the first example of considerations in the complexity of deep inference yielding new results for more mainstream systems in proof complexity. Unlike the previous two results our proofs of the weak pigeonhole principle rely crucially on the fact that the proofs permuting threshold arguments we constructed have flows of polylogarithmic length. The basic idea is to begin with formulae *approximating* threshold functions and bound how much worse the approximation develops as the interleaving and transposition arguments of Sect. 3.1 are applied.

5.3.1 Approximating Threshold Functions

It is not quite correct to call the formulae we define below as ε-approximators of threshold functions, since in fact they output incorrectly on a large proportion of inputs. Rather they output 1 just if the actual threshold is within some predetermined factor of the threshold being measured. The tradeoff is that we are able to define monotone formulae that are much smaller than the usual threshold formulae we have used until now.

Definition 42 (Threshold Approximators)**.** Let $|\boldsymbol{a}| = |\boldsymbol{b}| = n$. We define the (p,q)-approximator $\mathsf{T}^n_k[p,q]$ of TH^n_k as follows,

$$\mathsf{T}^n_k[p,q](\boldsymbol{a},\boldsymbol{b}) = \bigvee_{i+j=p} \mathsf{T}^n_{\frac{ik}{q}}[p,q](\boldsymbol{a}) \wedge \mathsf{T}^n_{\frac{jk}{q}}[p,q](\boldsymbol{b})$$

where we assume that k is some power of q and n is a power of 2, for example by adding a string of $\top$s and $\bot$s of appropriate format to the arguments.[7]

It is not easy to understand the semantics of these approximators, and in the next section we provide solely proof-theoretic arguments rather than semantic intuition. We do, however, make the following observations, provable by straightforward inductions.

Observation 43. *We have the following properties,*

1. $\mathsf{TH}^n_k \Rightarrow \mathsf{T}^n_k[p,q]$ *for all* $p < q$.
2. $\mathsf{T}^n_k[p,q] \Rightarrow \mathsf{TH}^n_{k(p/q)^{\log n}}$.
3. $|\mathsf{T}^n_k[p,q](\boldsymbol{a})| = O(p^{\log n})$

where $\Rightarrow$ *denotes logical implication.*

5.3.2 Manipulating Arguments in Threshold Approximators

In this section we return to the derivations proved in Sect. 3.1 on interleaving and transposing arguments of a formula. Since the approximators we now consider do not exactly compute threshold functions, they are no longer symmetric and so similar derivations cannot be constructed. Rather we show that witnessing certain permutations requires a bounded deterioration in the accuracy of the approximation. Ultimately we will choose an initial approximation that is accurate enough to ensure that this deterioration does not become too excessive.

We omit proofs in this section due to space restrictions, but state intermediate results to show how the approximation deteriorates with various permutations. Full proofs, in particular of the crucial Lemma 45, can be found in an extended version of this paper [14].

The following is proved by a straightforward induction.

Lemma 44. *For* $p \geq p', k \geq k'$ *there are normal derivations,*

$$\begin{array}{c}\mathsf{T}^n_k[p,q](\boldsymbol{a})\\ \|\\ \mathsf{T}^n_{k'}[p',q](\boldsymbol{a})\end{array}$$

of size $p^{O(\log n)}$.

Lemma 45. *There are normal derivations,*

$$\begin{array}{c}\mathsf{T}^n_k[p,q](\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d})\\ \|\\ \mathsf{T}^n_k[p-1,q](\boldsymbol{a},\boldsymbol{c},\boldsymbol{b},\boldsymbol{d})\end{array}$$

of size $p^{O(\log n)}$.

The following result has proof similar to that of Lemma 20, using the above lemma in the inductive steps to measure the deterioration of the approximator.

Proposition 46. *There are monotone derivations,*

$$\begin{array}{c}\mathsf{T}^n_k[p,q](\boldsymbol{a},\boldsymbol{b})\\ \|\\ \mathsf{T}^n_k[p-\log n,q](\boldsymbol{a} \mathbin{|||} \boldsymbol{b})\end{array}$$

of size $p^{O(\log n)}$ *whose flows have length* $O(\log n)$ *and width* $O(p)$.

[7] This increases the number of arguments by at most multiplication by $2q$.

The following result has proof similar to that of Thm. 22.

Theorem 47. *There are monotone derivations,*

$$\begin{array}{c}\mathsf{T}^n_k[p,q](\boldsymbol{X})\\ \|\\ \mathsf{T}^n_k[p-\log^2 n,q](\boldsymbol{X}^{\intercal})\end{array}$$

of size $p^{O(\log n)}$ *whose flows have length* $O(\log^2 n)$ *and width* $O(p)$.

5.3.3 From Approximators to the Weak Pigeonhole Principle

Recall the definition of PHP^m_n, where m denotes an arbitrary number of pigeons greater than the number of holes n, and define LPHP^m_n and RPHP^n_m analogously to Dfn. 23. In this section we essentially mimic the results of Sect. 3.2 to complete our proofs of the weak pigeonhole principle.

First we will need the following well known result whose proof follows, for example, by consideration of the inclusion-exclusion principle in the binomial expansion.

Proposition 48. *For* $\varepsilon \leq 1$ *we have that* $(1-\varepsilon)^k \geq 1 - \varepsilon k$.

The following result has proof similar to that of Thm. 25.

Lemma 49. *For* $q > p$ *and* $k > \frac{n}{(p/q)^{\log n}}$ *there are normal derivations,*

$$\begin{array}{c}\mathsf{LPHP}^m_n\\ \|\\ \mathsf{T}^{mn}_m[p,q](p_{ij})\end{array} \quad , \quad \begin{array}{c}\mathsf{T}^{mn}_k[p,q](p_{ij})^{\intercal}\\ \|\\ \mathsf{RPHP}^m_n\end{array}$$

of size $p^{O(\log n)}$.

Theorem 50. *For* $\varepsilon = 1/\log^{\Omega(1)} n$ *there are monotone derivations,*

$$\begin{array}{c}\mathsf{LPHP}^n_{(1-\varepsilon)n}\\ \|\\ \mathsf{RPHP}^n_{(1-\varepsilon)n}\end{array}$$

of size $n^{O(\log\log n)}$*, width* $O(\log n)$ *and length* $O(\log^2 n)$.

Proof. For $\varepsilon = 1/\log^d n$, choose $q = 3\log^{d+3} n$ and $p = q-1$. Since $\varepsilon > \frac{1}{q}$, there is a trivial derivation from $\mathsf{LPHP}^n_{(1-\varepsilon)n}$ to $\mathsf{LPHP}^n_{(1-\frac{1}{q})n}$ in $\mathsf{w}{\downarrow}$, and by chaining this to the derivations from Lemmata 49 and 47 we obtain monotone derivations from $\mathsf{LPHP}^n_{(1-\varepsilon)n}$ to $\mathsf{T}^{(1-\frac{1}{q})n^2}_n[p-\log^2 n,q](p_{ij})^{\intercal}$.

We now need to check that $n > \frac{(1-\varepsilon)n}{((p-\log^2 n)/q)^{\log n}}$ before applying Lemma 49. Now we have that,

$$\begin{array}{rcl}\left(\frac{p-\log^2 n}{q}\right)^{\log n} & = & \left(\frac{3\log^{d+3} n - \log^2 n - 1}{3\log^{d+3} n}\right)^{\log n}\\ & = & \left(1 - \frac{\log^2 n + 1}{3\log^{d+3} n}\right)^{\log n}\\ & \geq & \left(1 - \frac{2}{3\log^{d+1} n}\right)^{\log n}\\ & \geq & 1 - \frac{2\log n}{3\log^{d+1} n} \geq 1 - \frac{2}{3\log^d n}\end{array}$$

by Prop. 48. Consequently we have that,

$$\frac{(1-\varepsilon)}{((p-\log^2 n)/q)^{\log n}} \leq \frac{1 - \frac{1}{\log^d n}}{1 - \frac{2}{3\log^d n}} < 1$$

giving monotone derivations from $\mathsf{LPHP}^n_{(1-\varepsilon)n}$ to $\mathsf{RPHP}^n_{(1-\varepsilon)n}$ by Lemma 49. From previous bounds, the size of these derivations is $p^{O(\log n)} = (\log n)^{O(\log n)} = n^{O(\log\log n)}$ as required. $\square$

Since the width of these derivations is $O(\log n)$ we also gain a minor improvement in the complexity of KS proofs of $\mathsf{PHP}^n_{(1-\varepsilon)n}$ over those appearing in Sect. 3.

Corollary 51. *There are* KS *proofs of* $\mathsf{PHP}^n_{(1-\varepsilon)n}$*, for* $\varepsilon = 1/\log^{\Omega(1)} n$*, of size* $n^{O(\log n \log\log n)}$.

6. Final Comments

We constructed explicit quasipolynomial-size proofs of the pigeonhole principle in KS, and generalized our techniques to further yield quasipolynomial-size proofs of the parity principle and quite strong variants of the weak pigeonhole principle. In particular the existence of $n^{O(\log\log n)}$-size monotone proofs of the most common variant, with $2n$ pigeons and n holes, are implied by our construction. We repeat that this is the first time when considerations in the complexity of deep inference proofs have led to improvements for systems in mainstream proof complexity.

The various proof structures used throughout this work are similar in concept and fairly uniform, and so it might be pertinent to design high-level tools to more easily manipulate deep inference proofs, respecting certain complexity properties. One such approach might be to design an associated theory of *bounded artithmetic*, as done for other propositional proof systems, e.g. the theory $I\Delta_0$ for bounded-depth Frege systems [23]. Work in this direction is ongoing.

A natural question is whether the methods used here could be generalized to yield a simulation of Frege proofs, as done for KS^+ in [10] [19]. That construction is also relies heavily on threshold formulae, however it is not clear how to restrict the length of flows in the same way as we did here.

Note that the proofs we have given, albeit $n^{O(\log^2 n)}$ so quasipolynomial in size, are not in polynomial correspondence with those constructed in [2] for the monotone sequent calculus, and so KS^+, which have smaller quasipolynomial size $n^{O(\log n)}$. In fact it is conjectured that there are polynomial-size proofs in KS^+, due to the more general conjecture that the monotone sequent calculus polynomially simulates the full sequent calculus over monotone sequents. Consequently we cannot rule out the possibility that proofs of PHP_n witness a superpolynomial separation between KS and KS^+.

References

[1] M. Ajtai. Parity and the pigeonhole principle. In S. Buss and P. Scott, editors, *Feasible Mathematics*, volume 9 of *Progress in Computer Science and Applied Logic*, pages 1–24. Birkhuser Boston, 1990. ISBN 978-0-8176-3483-4. . URL `http://dx.doi.org/10.1007/978-1-4612-3466-1_1`.

[2] A. Atserias, N. Galesi, and R. Gavaldà. Monotone proofs of the pigeon hole principle. *Mathematical Logic Quarterly*, 47(4):461–474, 2001. ISSN 1521-3870. . URL `http://www.lsi.upc.edu/~atserias/papers/php/proof9.pdf`.

[3] A. Atserias, N. Galesi, and P. Pudlák. Monotone simulations of non-monotone proofs. *Journal of Computer and System Sciences*, 65(4): 626–638, 2002.

[4] P. Beame and T. Pitassi. An exponential separation between the parity principle and the pigeonhole principle. pages 195–228, 1996.

[5] K. Brünnler. Two restrictions on contraction. *Logic Journal of the IGPL*, 11(5):525–529, 2003. `http://www.iam.unibe.ch/~kai/Papers/RestContr.pdf`.

[6] K. Brünnler. *Deep Inference and Symmetry in Classical Proofs*. Logos Verlag, Berlin, 2004. `http://www.iam.unibe.ch/~kai/Papers/phd.pdf`.

[7] K. Brünnler and R. McKinley. An algorithmic interpretation of a deep inference system. In I. Cervesato, H. Veith, and A. Voronkov, editors, *LPAR 2008*, volume 5330 of *Lecture Notes in Computer Science*, pages 482—496. Springer-Verlag, 2008. `http://www.iam.unibe.ch/~kai/Papers/2008aidis.pdf`.

[8] K. Brünnler and A. F. Tiu. A local system for classical logic. Technical report, 2001. `http://www.iam.unibe.ch/~kai/Papers/lcl-lpar.pdf`.

[9] P. Bruscoli and A. Guglielmi. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*, 10(2):1–34, 2009. Article 14. `http://cs.bath.ac.uk/ag/p/PrComplDI.pdf`.

[10] P. Bruscoli, A. Guglielmi, T. Gundersen, and M. Parigot. A quasipolynomial cut-elimination procedure in deep inference via atomic flows and threshold formulae. In E. M. Clarke and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-16)*, volume 6355 of *Lecture Notes in Computer Science*, pages 136–153. Springer-Verlag, 2010. . URL `http://cs.bath.ac.uk/ag/p/QPNDI.pdf`.

[11] S. R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52(4):916–927, 1987.

[12] A. Das. On the proof complexity of cut-free bounded deep inference. In K. Brünnler and G. Metcalfe, editors, *Tableaux 2011*, volume 6793 of *Lecture Notes in Artificial Intelligence*, pages 134–148. Springer-Verlag, 2011. . URL `http://www.anupamdas.com/items/PrCompII/ProofComplexityBoundedDI.pdf`.

[13] A. Das. Complexity of deep inference via atomic flows. In S. B. Cooper, A. Dawar, and B. Löwe, editors, *Computability in Europe*, volume 7318 of *Lecture Notes in Computer Science*, pages 139–150. Springer-Verlag, 2012. `http://www.anupamdas.com/items/RelComp/RelComp.pdf`.

[14] A. Das. On the pigeonhole and related principles in deep inference and monotone systems. `http://anupamdas.com/items/WeakMonProofsPHP/WeakMonProofsPHP-Extended.pdf`, 2014.

[15] E. W. Dijkstra. The undeserved status of the pigeon-hole principle. Mar. 1991. URL `http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1094.PDF`.

[16] A. Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1–64, 2007. `http://cs.bath.ac.uk/ag/p/SystIntStr.pdf`.

[17] A. Guglielmi and T. Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9): 1–36, 2008. `http://www.lmcs-online.org/ojs/viewarticle.php?id=341`.

[18] A. Guglielmi, T. Gundersen, and M. Parigot. A proof calculus which reduces syntactic bureaucracy. In C. Lynch, editor, *RTA 2010*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 135–150. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2010. `http://drops.dagstuhl.de/opus/volltexte/2010/2649`.

[19] E. Jeřábek. Proof complexity of the cut-free calculus of structures. *Journal of Logic and Computation*, 19(2):323–339, 2009. `http://www.math.cas.cz/~jerabek/papers/cos.pdf`.

[20] E. Jeřábek. A sorting network in bounded arithmetic. *Annals of Pure and Applied Logic*, 162(4):341–355, 2011.

[21] E. Jeřábek. Proofs with monotone cuts. *Mathematical Logic Quarterly*, 58(3):177–187, 2012.

[22] J. Krajíček, P. Pudlák, and A. 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. ISSN 1098-2418. . URL `http://dx.doi.org/10.1002/rsa.3240070103`.

[23] J. Paris and A. Wilkie. δ_0 sets and induction. *Open Days in Model Theory and Set Theory, W. Guzicki, W. Marek, A. Pelc, and C. Rauszer, eds*, pages 237–248, 1981.

[24] T. Pitassi, P. Beame, and R. Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993. ISSN 1016-3328. URL `http://dx.doi.org/10.1007/BF01200117`. 10.1007/BF01200117.

[25] A. Razborov. Proof complexity of pigeonhole principles. In *Developments in Language Theory*, pages 203–206. Springer, 2002.

[26] L. Straßburger. Extension without cut. *Annals of Pure and Applied Logic*, 163(12):1995–2007, 2012. . URL `http://www.lix.polytechnique.fr/~lutz/papers/psppp.pdf`.