Hardness of Parameterized Resolution

Olaf Beyersdorff [1] * Nicola Galesi [2] † Massimo Lauria [2]

[1] *Institut für Informatik, Humboldt-Universität zu Berlin, Germany*
[2] *Dipartimento di Informatica, Sapienza Università di Roma, Italy*

Abstract

Parameterized Resolution and, moreover, a general framework for parameterized proof complexity was introduced by Dantchev, Martin, and Szeider [16] (FOCS'07). In that paper, Dantchev et al. show a complexity gap in tree-like Parameterized Resolution for propositional formulas arising from translations of first-order principles.

We broadly investigate Parameterized Resolution obtaining the following main results:

- We introduce a purely combinatorial approach to obtain lower bounds to the proof size in tree-like Parameterized Resolution. For this we devise a new asymmetric Prover-Delayer game which characterizes proofs in (parameterized) tree-like Resolution. By exhibiting good Delayer strategies we then show lower bounds for the pigeonhole principle as well as the order principle.
- Interpreting a well-known FPT algorithm for vertex cover as a DPLL procedure for Parameterized Resolution, we devise a proof search algorithm for Parameterized Resolution and show that tree-like Parameterized Resolution allows short refutations of all parameterized contradictions given as bounded-width CNF's.
- We answer a question posed by Dantchev, Martin, and Szeider in [16] showing that dag-like Parameterized Resolution is not fpt-bounded. We obtain this result by proving that the pigeonhole principle requires proofs of size $n^{\Omega(k)}$ in dag-like Parameterized Resolution. For this lower bound we use a different Prover-Delayer game which was developed for Resolution by Pudlák [27].

1 Introduction

Resolution is a well-known refutation system for unsatisfiable formulas in conjunctive normal form based on the single Resolution rule $\frac{\{x\}\cup C \quad \{\neg x\}\cup D}{C\cup D}$, where C, D are clauses and x is a variable. It was introduced by Blake [10] and since the work of Robinson [30] and Davis, Putnam, Logemann, and Loveland [17,18] has been highly employed in proof search and automated theorem proving. More recently, the study of Resolution has received a lot of attention and gained great significance in at least two important fields of computer science: (1) *Proof complexity*, where Resolution is the most intensively investigated proof system [2,6,9,11,14,24,34] and the study of lower bounds for proof lengths in this system has opened the way to lower bounds in much stronger proof systems [7,32]; (2) *Algorithms for the satisfiability problem* of CNF formulas, where the DPLL algorithm [4,17] is the core of the most important and modern algorithms employed for the satisfiability problem [4,5]. Running DPLL on unsatisfiable formulas produces

*This work was done while the first author was visiting Sapienza University of Rome under support of DFG grant KO 1053/5–2.

†Supported by grant "Limiti di compressione in combinatoria e complessità computazionale" by Sapienza University Rome.

Resolution refutations in the simple form of a tree, thus Resolution proof lengths are connected with the running time of DPLL procedures.

Very recently, Dantchev, Martin, and Szeider [16] introduced the proof system of *Parameterized Resolution*, which is a Resolution system in the context of *parameterized proof complexity*, an extension of the proof complexity approach of Cook and Reckhow to parameterized complexity which was initiated in the same work [16]. Parameterized complexity is mainly motivated by the quest for efficient algorithms which solve optimization problems [19,21,26]. Since Resolution is very important for SAT solving, Parameterized Resolution combines these two approaches, and its investigation might provide new insights into proof search for tractable fragments of classically hard problems. Some results in this directions are already outlined in the work of Gao [23] where he analyzes the effect of the standard DPLL algorithm on the problem of weighted satisfiability for random d-CNF. However, the study of Parameterized Resolution and our understanding of the possible implications for SAT-solving algorithms are still at a very early stage.

In this work we broadly study the Parameterized Resolution system proving hardness results for the lengths of proofs in both the tree-like and the dag-like version (thus answering a question posed in [16]). We devise a proof search algorithm that shows the strength of these systems over classical Resolution and yields some insight into questions in parameterized complexity. We start with a short overview of parameterized proof complexity and then discuss in more detail our contributions, their importance, how they extend the work [16] and previous work in Resolution, and future perspectives opened.

1.1 Parameterized Complexity and Parameterized Proof Complexity

Parameterized complexity is a branch of complexity theory where problems are analyzed in a finer way than in the classical approach: we say that a problem is *fixed-parameter tractable* (FPT) with parameter k if it can be solved in time $f(k)n^{O(1)}$ for some computable function f of arbitrary growth. In this setting, classically intractable problems may have efficient solutions for small choices of the parameter, even if the total size of the input is large. Parameterized complexity also has completeness theory. Many parameterized problems that appear to be not fixed-parameter tractable have been classified as being complete under fpt-reductions for complexity classes in the so-called weft hierarchy $\mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \mathsf{W}[3] \subseteq \ldots$

Consider the problem Weighted CNF Sat of finding a truth assignment of Hamming weight at most k that satisfies all clauses of a formula in conjunctive normal form. Many parameterized combinatorial problems can be naturally encoded in Weighted CNF Sat: finding a vertex cover of size at most k, finding a clique of size k, or finding a dominating set of size at most k. In the theory of parameterized complexity, the hardness of the Weighted CNF Sat problem is reflected by the fact that it is $\mathsf{W}[2]$-complete. Parameterized complexity has a very well-developed and deep theory and, as for the classical case, there are many open problems concerning the separation of parameterized complexity classes as FPT and $\mathsf{W}[\mathsf{P}]$ (see [19,22,26] for a complete treatment of the field).

Recently, Dantchev, Martin, and Szeider [16] initiated the study of *parameterized proof complexity*. After considering the notions of propositional *parameterized tautologies* and *fpt-bounded* proof systems, they laid the foundations to study complexity of proofs in a parameterized setting. The problem Weighted CNF Sat leads to parameterized contradictions:

Definition 1 (see [16]). *A* parameterized contradiction *is a pair* (F, k) *consisting of a propositional formula* F *and* $k \in \mathbb{N}$ *such that* F *has no satisfying assignment of weight* $\leq k$. *We denote the set of all parameterized contradictions by* PCon.

The notions of a parameterized proof system and of fpt-bounded proof systems were also developed in [16] (see Section 2 for a discussion):

Definition 2 (Dantchev et al. [16])**.** *A* parameterized proof system *for a parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ is a function $P : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that $rng(P) = L$ and $P(x, k)$ can be computed in time $O(f(k)|x|^{O(1)})$ with some computable function f.*

The system P is fpt-bounded *if there exist computable functions s and t such that every $(x, k) \in L$ has a P-proof (y, k') with $|y| \leq s(k)|x|^{O(1)}$ and $k' \leq t(k)$.*

The main motivation behind the work of [16] was that of generalizing the classical approach of Cook and Reckhow to the parameterized case and working towards a separation of parameterized complexity classes as FPT and W[P] by techniques developed in proof complexity. In fact, we obtain an analogous result to the well-known Cook-Reckhow theorem from [15]:

Theorem 3. *A parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ has an fpt-bounded proof system if and only if $L \in$ para-NP.*

Parameterized Resolution is a refutation system for the set of parameterized contradictions. Given a set of clauses F in variables $x_1, \ldots, x_n$ with $(F, k) \in$ PCon, a *Parameterized Resolution refutation* of (F, k) is a Resolution refutation of $F \cup \{\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}} \mid 1 \leq i_1 < \cdots < i_{k+1} \leq n\}$. Thus, in Parameterized Resolution we have built-in access to all parameterized clauses of the form $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. All these clauses are available in the system, but when measuring the size of a derivation we only count those which appear in the derivation.

Dantchev et al. [16] prove an extension of Riis' *gap theorem* [29], getting a separation for the complexity of tree-like Parameterized Resolution proofs for formulas arising from propositional encodings of first-order principles $\mathcal{P}$, that uniquely depends on $\mathcal{P}$ having or not infinite models.

1.2 Our Contributions

Dantchev, Martin, and Szeider [16] proved that tree-like Parameterized Resolution is not ftp-bounded. However, their lower bound technique only works for formulas arising from propositional encodings of first-order principles having infinite models.

As our first main contribution (Section 3) we devise a purely combinatorial approach, based on a new *asymmetric Prover-Delayer game*, to characterize proof size in tree-like Parameterized Resolution. In particular, we use our characterization to prove lower bounds in this system. Our game is inspired by the Prover-Delayer game of Pudlák and Impagliazzo [28], which is one of the canonical tools to study lower bounds in tree-like Resolution [8,28] and tree-like $Res(k)$ [20]. As we argue in next subsection, our game is more general than the one of Pudlák and Impagliazzo (which is a "constant" case of our game) and can be used also for standard Resolution. Using this game, lower bounds to the proof size in tree-like Parameterized Resolution immediately follow from good strategies for the Delayer. We provide such strategies for the case of the pigeonhole principle (Theorem 9) and for the case of a (partial) ordering principle (Theorem 16) obtaining the hardness of refuting these two principles in tree-like Parameterized Resolution.

As our second contribution we investigate classes of parameterized contradictions that have short refutations in Parameterized Resolution. The notion of *efficient kernelization* plays an important role in the theory of parameterized complexity to design fpt-algorithms. Here we propose a notion of kernel for parameterized proof complexity. We observe that if a formula has a kernel, then it can be efficiently refuted in tree-like Parameterized Resolution. As an immediate consequence several examples of formulas hard for tree-like Resolution are instead efficiently refutable in the parameterized case: pebbling contradictions, linear ordering principles, graph pigeonhole principles, and colorability principles. But sometimes a kernel of a formula is not

explicit or immediate to find. In Theorem 11 we prove that contradictions of bounded width have a kernel and thus very efficient tree-like Parameterized Resolution refutations. This implies that formulas like random 3-CNF's and (negations of) Tseitin tautologies are efficiently refutable in tree-like Parameterized Resolution, even though they are known to be hard for systems even stronger than (dag-like) Resolution.

Our third and most important contribution answers a question raised by Dantchev, Martin, and Szeider [16, Section 4]: What is the complexity of the pigeonhole principle (PHP) in dag-like Parameterized Resolution? Clearly this question becomes even more interesting in the light of our previous upper bounds. Using a certain encoding of the parameterized axioms—based on the same pigeonhole principle—one can get efficient proofs of PHP in Parameterized Resolution (Proposition 17 in [16]). In contrast, we show that dag-like Parameterized Resolution is not fpt-bounded by proving that PHP requires proofs of size $n^{\Omega(k)}$ (Theorem 21). Again for this lower bound we use an interpretation of proofs in Parameterized Resolution as games. Here we employ a slight modification of Pudlák's game for Resolution lower bounds as devised in [27], although, as we argue in the next subsection, the analysis for the lower bound of PHP in Resolution does not suffice for proving lower bounds in Parameterized Resolution. Our result on dag-like Parameterized Resolution proofs for PHP represents the next step in the program proposed by the work of [16] that approaches the separation of FPT from W[2] as in the Cook-Reckhow program of separating NP from coNP for classical proof complexity.

1.3 Techniques and Proof Methods

1.3.1 Asymmetric Prover-Delayer Game and Information Content of Proofs

The original Prover-Delayer game, developed by Pudlák and Impagliazzo [28], arises from the well-known fact [25] that a tree-like Resolution proof for a formula F can be viewed as a decision tree which solves the search problem of finding a clause of F falsified by a given assignment. In the game, Prover queries a variable and Delayer either gives it a value or leaves the decision to Prover and receives *one* point. The number of Delayer's points at the end of the game is then proportional to the height of the proof tree. It is easy to argue that showing lower bounds by this game only works if (the graph of) every tree-like Resolution refutation contains a balanced sub-tree as a minor, and the height of that sub-tree then gives the size lower bound. This is clearly a weakness of the method, for instance in case of the (non-parameterized) pigeonhole principle, whose tree-like Resolution refutations are of size roughly $n!$ but only contain balanced sub-trees of height n.

Our new game, in contrast, assigns points to the Delayer asymmetrically ($\log c_0$ and $\log c_1$) according to two functions c_0 and c_1 which depend on the principle, the variable queried, and the current partial assignment. In fact, the original Prover-Delayer game of [28] is just a $(2, 2)$-asymmetric game. Proving lower bounds in our new game, i. e., devising good Delayer strategies, entails first of all to finding suitable functions c_0 and c_1. Functions c_0 and c_1 can be interpreted in terms of *information content* of tree-like Resolution refutations. The points which Delayer scores in one round should be proportional to the fraction of the current refutation that Prover can avoid to check by deciding a value for the variable. This is easily understandable in the case of the original game: the only good strategy for Prover to set a variable is choosing the value that allows him to proceed the game in the smallest of the two sub-trees of the current refutation which is in fact of size smaller than $1/2$ of the current refutation size.

1.3.2 Proof Search and Parameterized DPLL

Gao [23] suggested to use a standard DPLL algorithm to find refutations of certain random parameterized d-CNF's. Here we prove that bounded width CNF's have a kernel and hence are efficiently refutable in tree-like Parameterized Resolution. The core of our quite simple argument is the interpretation of a classical parameterized algorithm for vertex cover as a DPLL procedure.

A vertex cover for a graph G is a set $C \subseteq V(G)$ such that for any $\{u, v\} \in E(G)$ either $u \in C$ or $v \in C$. To determine whether G has a vertex cover of size at most k there is a well-known [19, Chapter 3] fixed parameter tractable algorithm (here the parameter is k). This algorithm is based on the following observation: if a vertex is not in C, then all its neighbors must be in C. The algorithm is a simple recursive procedure which focuses on an arbitrary vertex u, and on its neighbors $v_1, \ldots, v_l$: if neither $G \setminus \{u\}$ has a vertex cover of size $k-1$ nor $G \setminus \{u, v_1, \ldots, v_l\}$ has a vertex cover of size $k-l$, then G has no vertex cover of size k.

This is easily interpretable as a parameterized DPLL procedure on the 2-CNF $F_G = \bigwedge_{\{u,v\} \in E(G)} (x_u \vee x_v)$ where x_u indicates whether $u \in C$. The DPLL procedure fixes an arbitrary variable x_u and splits on it. When $x_u = 1$, then the DPLL algorithm proceeds with analyzing $F_G \restriction_{x_u=1}$ which is equal to $F_{G \setminus \{u\}}$. When $x_u = 0$, then $x_{v_1} = 1, \ldots, x_{v_l} = 1$ by unit propagation. Thus the DPLL proceeds on formula $F_G \restriction_{\{x_u=0, x_{v_1}=1, \ldots, x_{v_l}=1\}} = F_{G \setminus \{u, v_1, \ldots, v_l\}}$. If at any point the DPLL has more than k variables set to one, it stops and backtracks. In Theorem 11 we extend this idea to a DPLL algorithm for bounded width formulas.

1.3.3 Proofs as Games in Dag-like Parameterized Resolution

In studying lower bounds for Parameterized Resolution one is immediately faced with the following observations that exclude some techniques used for proving lower bounds in classical Resolution. First, Weighted CNF Sat is not resilient to restrictions in the sense that if we apply a restriction to a parameterized contradiction (F, k) we obtain (F', k') with $k' \leq k$. Then, either we use restrictions with only 0's which does not make sense, or we restrict k too much. The *width* lower bound method for Resolution [6, 9], which is based on restrictions, can therefore be excluded from the bag of techniques to prove lower bounds in Parameterized Resolution. Also one has to take into account that constant width CNF's have efficient tree-like Parameterized Resolution refutations. This implies that any technique employed to prove lower bounds in Parameterized Resolution for formulas of high initial width must not be preserved under the use of *extension variables* which reduce the width of a formula to a constant.

We decide to study the complexity of the pigeonhole principle in Parameterized Resolution employing a refined analysis of the original proof by Haken [24], but using the interpretation of his method as a game given by Pudlák in his work [27]. Following Pudlák [27] we can interpret a refutation of (F, k) in Parameterized Resolution as the following interactive process: Delayer claims to know an assignment α (a partial matching in the case of PHP) with at most k variables set to 1 and which satisfies F. The Prover queries the Delayer for variable values, can save such values in his memory (the "record") and can forget some of them at will. Prover wants to expose Delayer lie by exhibiting either an initial or a parameterized clause which is falsified by the assignment. Since there is a 1-1 correspondence between memory records and clauses in the proof we want to show that Prover always needs a high $(n^{\Omega(k)})$ number of records to expose the Delayer. This will be proved showing that to win the game Prover always has to write down a sufficient part of the assignment α in his record.

To show exponential lower bounds for general Resolution, Pudlák analyses the probability that a given Prover record is *compatible* with a chosen Delayer strategy (the assignment α), i. e., he bounds the probability that a record can appear during some game. However, this analysis

does not suffice to prove lower bounds in Parameterized Resolution. Instead of *compatibility*, we therefore focus on the stronger notion that a given record is *winning* in a game and we show that each Prover record can only be winning against a small fraction of Delayer strategies.

1.4 Organization of the Paper

The remaining part of the paper is organized as follows. Section 2 contains all preliminary notions and definitions concerning fixed-parameter tractability, parameterized proof systems, and Parameterized Resolution. In Section 3 we define our asymmetric Prover-Delayer game and establish its precise relation to proof size in tree-like Parameterized Resolution. We also show the tree-like lower bound for the pigeonhole principle. Section 4 concentrates on upper bounds: we introduce the notion of a kernel and prove that parameterized contradictions with kernels and of bounded width have efficient tree-like refutations. Section 5 is devoted to the analysis of different ordering principles. We prove that a formulation of linear ordering without a kernel still has efficient tree-like refutations and, using the prover-delayer game, we obtain hardness of a (partial) ordering principle for parameterized tree-like resolution. In Section 6 we show that general Parameterized Resolution is not fpt-bounded by proving a lower bound for the pigeon hole principle. Finally, in Section 7 we discuss open problems.

2 Parameterized Proof Complexity

2.1 Fixed-Parameter Tractability

A *parameterized language* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance (x, k), we call k the parameter of (x, k). A parameterized language L is *fixed-parameter tractable* if L has a deterministic decision algorithm running in time $f(k)|x|^{O(1)}$ for some computable function f. The class of all fixed-parameter tractable languages is denoted by FPT.

Besides FPT there is a wealth of complexity classes containing problems which are not believed to be fixed-parameter tractable. The most prominent classes lie in the *weft hierarchy* forming a chain

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{para\text{-}NP} \ .$$

The classes of the weft hierarchy are usually defined as the closure of a canonical problem under fpt-reductions. For $\mathsf{W}[2]$ this canonical problem is Weighted CNF Sat containing instances (F, k) with a propositional formula F in CNF and a parameter $k \in \mathbb{N}$. Weighted CNF Sat asks whether F has a satisfying assignment of weight k, where the weight of an assignment α, denoted $w(\alpha)$, is the number of variables that α assigns to 1. Instead of asking for an assignment α with $w(\alpha) = k$ we can also ask for α with $w(\alpha) \leq k$ and still get a $\mathsf{W}[2]$-complete problem (cf. [16] and the appendix). Like in the classical duality between tautologies and satisfiability, we obtain a complete problem for $\mathsf{coW}[2]$:

Definition 1 (Dantchev, Martin, Szeider [16])**.** *A* parameterized contradiction *is a pair (F, k) consisting of a propositional formula F and $k \in \mathbb{N}$ such that F has no satisfying assignment of weight $\leq k$. We denote the set of all parameterized contradictions by* PCon.

For an in-depth treatment of notions from parameterized complexity we refer to the monographs [19, 22, 26].

2.2 Parameterized Proof Systems

We start with the general definition of a parameterized proof system of Dantchev, Martin, and Szeider [16].

Definition 2 (Dantchev, Martin, Szeider [16]). *A* parameterized proof system *for a parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ is a function $P : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that $rng(P) = L$ and $P(x,k)$ can be computed in time $O(f(k)|x|^{O(1)})$ with some computable function f.*

In this definition, there are two parameters: one stemming from the parameterized language, but there is also a parameter in the proof. Verification of proofs then proceeds in fpt-time in the proof parameter. We contrast this with the classical concept of Cook and Reckhow where proofs are verified in polynomial time:

Definition 4 (Cook, Reckhow [15]). *A* proof system *for a language $L \subseteq \Sigma^*$ is a polynomial-time computable function $P : \Sigma^* \to \Sigma^*$ with $rng(P) = L$.*

This framework can also be applied to parameterized languages $L \subseteq \Sigma^* \times \mathbb{N}$. Thus, in contrast to the parameterized proof systems from Definition 2, we say that a *proof system for the parameterized language* L is just a polynomial-time computable function $P : \Sigma^* \to \Sigma^* \times \mathbb{N}$ with $rng(P) = L$. The difference to Definition 2 is that proofs are now verified in polynomial time. We will argue in Theorem 3 that this weaker and simpler notion is in fact equivalent to the notion of parameterized proof systems from Definition 2 when considering lengths of proofs.

For lengths of proofs it is appropriate to adjust the notion of short proofs to the parameterized setting. This was formalized by Dantchev, Martin, and Szeider as follows:

Definition 5 (Dantchev, Martin, Szeider [16]). *A parameterized proof system P for a parameterized language L is* fpt-bounded *if there exist computable functions f and g such that every $(x,k) \in L$ has a P-proof (y,k') with $|y| \leq f(k)|x|^{O(1)}$ and $k' \leq g(k)$.*

Again, if we use polynomial-time computable proof systems for parameterized languages, this definition simplifies a bit as follows:

Definition 6. *A proof system P for a parameterized language L is* fpt-bounded *if there exists a computable function f such that every $(x,k) \in L$ has a P-proof of size at most $f(k)|x|^{O(1)}$.*

We now want to determine which parameterized languages admit fpt-bounded proof systems. Recall that by the theorem of Cook and Reckhow [15], the class of all languages with polynomially bounded proof systems coincides with NP. To obtain a similar result in the parameterized world, we use the following parameterized version of NP.

Definition 7 (Flum, Grohe [21]). *The class* para-NP *contains all parameterized languages which can be decided by a nondeterministic Turing machine in time $f(k)|x|^{O(1)}$ for some computable function f.*

The following result is a direct analogue of the classical theorem of Cook and Reckhow [15] for the parameterized setting. Moreover, it shows the equivalence of our two notions for proof systems for parameterized languages with respect to the existence of fpt-bounded systems.

Theorem 3. *Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized language. Then the following statements are equivalent:*

1. *There exists an fpt-bounded proof system for L.*
2. *There exists an fpt-bounded parameterized proof system for L.*
3. *$L \in$ para-NP.*

Proof. For the implication $1 \Rightarrow 2$, let P be an fpt-bounded proof system for L. Then the system P' defined by $P'(y,k) = P(y)$ is an fpt-bounded parameterized proof system for L.

For the implication $2 \Rightarrow 3$, let P be an fpt-bounded parameterized proof system for L such that every $(x,k) \in L$ has a P-proof (y,k') with $|y| \leq f(k)p(|x|)$ and $k' \leq g(k)$ for some computable functions f, g and some polynomial p. Let M be a Turing machine computing P in time $h(k)q(n)$ with computable h and a polynomial q. Then $L \in \mathsf{para\text{-}NP}$ by the following algorithm: on input (x,k) we guess a proof (y,k') with $|y| \leq f(k)p(|x|)$ and $k' \leq g(k)$. Then we verify that $P(y,k') = (x,k)$ in time $h(k')q(|y|)$ which by the choice of (y,k) yields an fpt running time. If the test is true, then we accept the input (x,k), otherwise we reject.

For the implication $3 \Rightarrow 1$, let $L \in \mathsf{para\text{-}NP}$ and let M be a nondeterministic Turing machine for L running in time $f(k)p(n)$ where f is computable and p is a polynomial. Then we define the following proof system P for L:

$$P(x,k,w) = \begin{cases} (x,k) & \text{if } w \text{ is an accepting computation of } M \text{ on input } (x,k) \\ (x_0,k_0) & \text{otherwise} \end{cases}$$

where $(x_0,k_0) \in L$ is some fixed instance. Apparently, P can be computed in polynomial time. Moreover, P is fpt-bounded as every $(x,k) \in L$ has a P-proof of size $O(f(k)p(|x|))$. □

As in items 1 and 2 of Theorem 3, the two concepts of proof systems for parameterized languages also turn out to be equivalent with respect to other notions, for instance when defining parameterized simulations or considering the existence of optimal proof systems.

2.3 Parameterized Resolution

A *literal* is a positive or negated propositional variable and a *clause* is a set of literals. The *width* of a clause is the number of its literals. A clause is interpreted as the disjunctions of its literals and a set of clauses as the conjunction of the clauses. Hence clause sets correspond to formulas in CNF. The *Resolution system* is a refutation system for the set of all unsatisfiable CNF. Resolution uses as its only rule the *Resolution rule*

$$\frac{\{x\} \cup C \qquad \{\neg x\} \cup D}{C \cup D}$$

for clauses C, D and a variable x. The aim in Resolution is to demonstrate unsatisfiability of a clause set by deriving the empty clause. If in a derivation every derived clause is used at most once as a prerequisite of the Resolution rule, then the derivation is called *tree-like*, otherwise it is *dag-like*. The *size* of a Resolution proof is the number of its clauses where multiple instances of the same clause are counted separately. Undoubtedly, Resolution is the most studied and best-understood propositional proof system (cf. [31]).

For the remaining part of this paper we will concentrate on *Parameterized Resolution* as introduced by Dantchev, Martin, and Szeider [16]. Parameterized Resolution is a refutation system for the set PCon of parameterized contradictions (cf. Definition 1). Given a set of clauses F in variables $x_1, \ldots, x_n$ with $(F,k) \in \mathrm{PCon}$, a *Parameterized Resolution refutation* of (F,k) is a Resolution refutation of

$$F \cup \{\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}} \mid 1 \leq i_1 < \cdots < i_{k+1} \leq n\} \ .$$

Thus, in Parameterized Resolution we have built-in access to all parameterized clauses of the form $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. All these clauses are available in the system, but when measuring the size of a derivation we only count those which appear in the derivation. Note that parameterized resolution is actually a proof system where verification proceeds in polynomial time.

As before, if refutations are tree-like we speak of *tree-like Parameterized Resolution*. As explained in [16], a tree-like Parameterized refutation of (F, k) can equivalently be described as a *boolean decision tree*. A boolean decision tree for (F, k) is a binary tree where inner nodes are labeled with variables from F and leafs are labeled with clauses from F or parameterized clauses $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. Each path in the tree corresponds to a partial assignment where a variable x gets value 0 or 1 according to whether the path branches left or right at the node labeled with x. The condition on the decision tree is that each path α must lead to a clause which is falsified by the assignment corresponding to α. Therefore, a boolean decision tree solves the *search problem* for (F, k) which, given an assignment α, asks for a clause falsified by α. It is easy to verify that each tree-like Parameterized Resolution refutation of (F, k) yields a boolean decision tree for (F, k) and vice versa, where the size of the Resolution proof equals the number of nodes in the decision tree.

3 Tree-like Lower Bounds via Asymmetric Prover-Delayer Games

Let $(F, k) \in \mathrm{PCon}$ where F is a set of clauses in n variables $x_1, \ldots, x_n$. We define a Prover-Delayer game: Prover and Delayer build a (partial) assignment to $x_1, \ldots, x_n$. The game is over as soon as the partial assignment falsifies either a clause from F or a parameterized clause $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$ where $1 \leq i_1 < \cdots < i_{k+1} \leq n$. The game proceeds in rounds. In each round, Prover suggests a variable x_i, and Delayer either chooses a value 0 or 1 for x_i or leaves the choice to the Prover. In this last case, if the Prover sets the value, then the Delayer gets some points. The number of points Delayer earns depends on the variable x_i, the assignment α constructed so far in the game, and two functions $c_0(x_i, \alpha)$ and $c_1(x_i, \alpha)$. More precisely, the number of points that Delayer will get is

$$\begin{array}{cl} 0 & \text{if Delayer chooses the value,} \\ \log c_0(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 0, and} \\ \log c_1(x_i, \alpha) & \text{if Prover sets } x_i \text{ to 1.} \end{array}$$

Moreover, the functions $c_0(x, \alpha)$ and $c_1(x, \alpha)$ are chosen in such a way that for each variable x and assignment α

$$\frac{1}{c_0(x, \alpha)} + \frac{1}{c_1(x, \alpha)} = 1 \tag{1}$$

holds. Let us call this game the (c_0, c_1)-game on (F, k).

The connection of this game to size of proofs in tree-like Parameterized Resolution is given by the next theorem:

Theorem 8. *Let (F, k) be a parameterized contradiction and let c_0 and c_1 be two functions satisfying (1) for all partial assignments α to the variables of F. If (F, k) has a tree-like Parameterized Resolution refutation of size at most S, then the Delayer gets at most $\log S$ points in each (c_0, c_1)-game played on (F, k).*

Proof. Let (F, k) be a parameterized contradiction in variables $x_1, \ldots, x_n$ and let Π be a tree-like Parameterized Resolution refutation of (F, k). Assume now that Prover and Delayer play a game on (F, k) where they successively construct an assignment α. Let α_i be the partial assignment constructed after i rounds of the game, i. e., α_i assigns i variables a value 0 or 1. By p_i we denote the number of points that Delayer has earned after i rounds, and by Π_{α_i} we denote the sub-tree of the decision tree of Π which has as its root the node reached in Π along the path specified by α_i.

We use induction on the number of rounds i in the game to prove the following claim:

$$|\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} \ .$$

To see that the theorem follows from this claim, let α be an assignment constructed during the game yielding p_α points to the Delayer. As a contradiction has been reached in the game, the size of Π_α is 1, and therefore by the inductive claim

$$1 \leq \frac{|\Pi|}{2^{p_\alpha}} \ ,$$

yielding $p_\alpha \leq \log |\Pi|$ as desired.

In the beginning of the game, Π_{α_0} is the full tree and the Delayer has 0 points. Therefore the claim holds.

For the inductive step, assume that the claim holds after i rounds and Prover asks for a value of the variable x in round $i+1$. If the Delayer chooses the value, then $p_{i+1} = p_i$ and hence

$$|\Pi_{\alpha_{i+1}}| \leq |\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} = \frac{|\Pi|}{2^{p_{i+1}}} \ .$$

If the Delayer defers the choice to the Prover, then the Prover uses the following strategy to set the value of x. Let $\alpha_i^{x=0}$ be the assignment extending α_i by setting x to 0, and let $\alpha_i^{x=1}$ be the assignment extending α_i by setting x to 1. Now, Prover sets $x = 0$ if $|\Pi_{\alpha_i^{x=0}}| \leq \frac{1}{c_0(x,\alpha_i)}|\Pi_{\alpha_i}|$, otherwise he sets $x = 1$. Because $\frac{1}{c_0(x,\alpha_i)} + \frac{1}{c_1(x,\alpha_i)} = 1$, we know that if Prover sets $x = 1$, then $|\Pi_{\alpha_i^{x=1}}| \leq \frac{1}{c_1(x,\alpha_i)}|\Pi_{\alpha_i}|$. Thus, if Prover's choice is $x = j$ with $j \in \{0, 1\}$, then we get

$$|\Pi_{\alpha_{i+1}}| = |\Pi_{\alpha_i^{x=j}}| \leq \frac{|\Pi_{\alpha_i}|}{c_j(x,\alpha_i)} \leq \frac{|\Pi|}{c_j(x,\alpha_i)2^{p_i}} = \frac{|\Pi|}{2^{p_i + \log c_j(x,\alpha_i)}} = \frac{|\Pi|}{2^{p_{i+1}}} \ .$$

This completes the proof of the induction. □

Notice that setting $c_0(x, \alpha) = c_1(x, \alpha) = 2$ for all variables x and partial assignments α, we get the game of Pudlák and Impagliazzo [28]. Suitably choosing functions c_0 and c_1 and defining a Delayer-strategy for the (c_0, c_1)-game we can prove a lower bound to the proof size in tree-like Parameterized Resolution. We will illustrate this for the *pigeonhole principle* PHP_n^{n+1} which uses variables $x_{i,j}$ with $i \in [n+1]$ and $j \in [n]$, indicating that pigeon i goes into hole j. PHP_n^{n+1} consists of the clauses

$$\bigvee_{j \in [n]} x_{i,j} \quad \text{for all pigeons } i \in [n+1]$$

and $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for all choices of distinct pigeons $i_1, i_2 \in [n+1]$ and holes $j \in [n]$. We prove that PHP_n^{n+1} is hard for tree-like Parameterized Resolution.

Theorem 9. *Any tree-like Parameterized Resolution refutation of* (PHP_n^{n+1}, k) *has size* $n^{\Omega(k)}$.

Proof. Let α be a partial assignment to the variables $\{x_{i,j} \mid i \in [n+1], j \in [n]\}$. Let $z_i(\alpha) = |\{j \in [n] \mid \alpha(x_{i,j}) = 0\}|$, i. e., $z_i(\alpha)$ is the number of holes already excluded by α for pigeon i. We define

$$c_0(x_{i,j}, \alpha) = \frac{n - z_i(\alpha)}{n - z_i(\alpha) - 1} \quad \text{and} \quad c_1(x_{i,j}, \alpha) = n - z_i(\alpha)$$

which apparently satisfies (1). We now describe Delayer's strategy in a (c_0, c_1)-game played on (PHP_n^{n+1}, k). If Prover asks for a value of $x_{i,j}$, then Delayer decides as follows:

set $\alpha(x_{i,j}) = 0$	if there exists $i' \in [n+1] \setminus \{i\}$ such that $\alpha(x_{i',j}) = 1$ or
	if there exists $j' \in [n] \setminus \{j\}$ such that $\alpha(x_{i,j'}) = 1$
set $\alpha(x_{i,j}) = 1$	if there is no $j' \in [n]$ with $\alpha(x_{i,j'}) = 1$ and $\lvert z_i(\alpha)\rvert \geq n - k$
let Prover decide	otherwise.

Intuitively, Delayer leaves the choice to Prover as long as pigeon i does not already sit in a hole, but there are at least k holes free for pigeon i, and there is no other pigeon sitting already in hole j. If Delayer uses this strategy, then clauses from PHP_n^{n+1} will not be violated in the game, i. e., a contradiction will always be reached on some parameterized clause. To verify this claim, let α be a partial assignment constructed during the game with $w(\alpha) \leq k$. Then, for every pigeon which has not been assigned to a hole yet, there are at least k holes where it could go (and of these only $w(\alpha)$ holes are already occupied by other pigeons). Thus α can be extended to a one-one mapping of exactly k pigeons to holes.

Therefore, at the end of the game exactly $k+1$ variables have been set to 1. Let us denote by p the number of variables set to 1 by Prover and let d be the number of 1's assigned by Delayer. As argued before $p + d = k + 1$. Let us check how many points Delayer earns in this game. If Delayer assigns 1 to a variable $x_{i,j}$, then pigeon i was not assigned to a hole yet and, moreover, there must be $n-k$ holes which are already excluded for pigeon i by α, i. e., for some $J \subseteq [n]$ with $|J| = n-k$ we have $\alpha(x_{i,j'}) = 0$ for all $j' \in J$. Most of these 0's have been assigned by Prover, as Delayer has only assigned a 0 to $x_{i,j'}$ when some other pigeon was already sitting in hole j', and there can be at most k such holes. Thus, before Delayer sets $\alpha(p_{i,j}) = 1$, she has already earned points for at least $n - 2k$ variables $x_{i,j'}$, $j' \in J$, yielding at least

$$\sum_{z=0}^{n-2k-1} \log \frac{n-z}{n-z-1} \;=\; \log \prod_{z=0}^{n-2k-1} \frac{n-z}{n-z-1} \;=\; \log \frac{n}{2k} \;=\; \log n - \log 2k$$

points for the Delayer. Let us note that because Delayer never allows a pigeon to go into more than one hole, she will really get the number of points calculated above for *every* of the d variables which she set to 1.

If, conversely, Prover sets variable $x_{i,j}$ to 1, then Delayer gets $\log(n - z_i(\alpha))$ points for this, but she also received points for most of the $z_i(\alpha)$ variables set to 0 before. Thus, in this case Delayer earns on pigeon i at least

$$\begin{aligned} & \log(n - z_i(\alpha)) + \sum_{z=0}^{z_i(\alpha)-k-1} \log \frac{n-z}{n-z+k-1} \\ = \;& \log(n - z_i(\alpha)) + \log \frac{n}{n - z_i(\alpha) + k} \\ = \;& \log n - \log \frac{n - z_i(\alpha) + k}{n - z_i(\alpha)} \\ \geq \;& \log n - \log k \end{aligned}$$

points. In total, Delayer gets at least

$$d(\log n - \log 2k) + p(\log n - \log k) \geq k(\log n - \log 2k)$$

points in the game. Applying Theorem 8, we obtain $(\frac{n}{2k})^k$ as a lower bound to the size of each tree-like Parameterized Resolution refutation of (PHP_n^{n+1}, k). □

By inspection of the above Delayer strategy it becomes clear that the lower bound from Theorem 9 also holds for the *functional pigeonhole principle* where in addition to the clauses from PHP_n^{n+1} we also include $\neg x_{i,j_1} \vee \neg x_{i,j_2}$ for all pigeons $i \in [n+1]$ and distinct holes $j_1, j_2 \in [n]$.

4 Kernels and Small Refutations

The notion of *efficient kernelization* plays an important role in the theory of parameterized complexity. A kernelization for a parameterized language L is a polynomial-time procedure $A : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that for each (x, k)

1. $(x, k) \in L$ if and only if $A(x, k) \in L$ and
2. if $A(x, k) = (x', k')$, then $k' \leq k$ and $|x'| \leq f(k)$ for some computable function f independent of $|x|$.

It is clear that if a parameterized language admits a kernelization, then the language is fixed-parameter tractable, but also the converse is true (cf. [22]). For parameterized proof complexity we suggest a similar notion of kernel for parameterized contradictions:

Definition 10. *A set* $\Gamma \subseteq \mathrm{PCon}$ *of parameterized contradictions has a* kernel *if there exists a computable function* f *such that every* $(F, k) \in \Gamma$ *has a subset* $F' \subseteq F$ *of clauses satisfying the following conditions:*

1. F' *contains at most* $f(k)$ *variables and*
2. (F', k) *is a parameterized contradiction.*

Note that a sequence of parameterized contradictions with a kernel of size $f(k)$ admits fpt-bounded tree-like Resolution refutations of size at most $2^{f(k)}$. Nevertheless, there are CNF's without a kernel, but with fpt-bounded refutations, for example $(x_1 \vee x_2 \vee \cdots \vee x_n) \wedge \neg x_1 \wedge \neg x_2 \wedge \cdots \wedge \neg x_n$.

We now give some examples of CNF's with kernels:

Pebbling contradictions. Fix a constant l and an acyclic connected directed graph G of constant maximum in-degree with a single sink vertex z. For any vertex v in G, let $Pred(v)$ be the set of immediate predecessors of v. For any v we use the propositional variables $x_1^v, \ldots, x_l^v$. The *pebbling contradiction* consists of the conjunction of the constraints

$$\left(\bigwedge_{u \in Pred(v)} (x_1^u \vee \ldots \vee x_l^u) \right) \longrightarrow x_1^v \vee \ldots \vee x_l^v$$

for any $v \in V(G)$ and constraints $\neg x_1^z, \neg x_2^z, \ldots, \neg x_l^z$. This means that for any source vertex s (which has an empty set of predecessors) one of the variables x_i^s is true. By induction this holds for every vertex in G, in particular also for the sink z contradicting the last l clauses. For constant l and constant maximum in-degree, the pebbling formula can be encoded as a CNF of polynomial size in the number of vertices of G.

To see that the pebbling contradictions have a kernel, consider the first $k+1$ vertices in a topological ordering of $V(G)$. The corresponding propagation formulas form a parameterized contradiction, because these formulas enforce $k+1$ true variables. For graphs of constant maximum in-degree, these formulas have $O(1)$ variables each, so their CNF encoding has size

$O(k)$ and constitutes a kernel. This is remarkable, because forms of pebbling tautologies are hard for tree-like Resolution in the non-parameterized setting [8].

Colorability. Fix a constant c and a graph G which is not c-colorable. The *c-coloring contradiction* is defined on variables $p_{u,j}$ where u is a vertex of G and j is one of the c colors. The CNF claims that G is c-colorable: (1) for any vertex u we have the clause $\bigvee_{1\leq j\leq c} p_{u,j}$ claiming that the vertex u gets a color; (2) for any edge $\{u,v\}$ in G and any color j, the clause $\neg p_{u,j} \vee \neg p_{v,j}$ claims that no pair of adjacent vertices gets the same color.

It is easy to see that the clauses of type (1) corresponding to any $k+1$ vertices form a kernel and thus have a short refutation. This contrasts with the fact that for random G and $c \geq 3$ this formula is hard in Resolution [3].

Graph pigeonhole principle. Fix G to be a bounded-degree bipartite graph, with the set of vertices partitioned into two sets: U of size $n+1$ (the pigeons) and V of size n (the holes). $PHP(G)$ is a variant of the pigeonhole principle where a pigeon can go only into a small set of holes as specified by the edges of G. If G is an expander, then the principle is hard for Resolution [9]. $PHP(G)$ consists of the following clauses: (1) for $u \in U$ we have $\bigvee_{v\in\Gamma(u)} p_{u,v}$; (2) for any $v \in V$ and any $u_1, u_2 \in \Gamma(v)$ we have the clause $\neg p_{u_1,v} \vee \neg p_{u_2,v}$. It is clear that $k+1$ clauses of type (1) constitute a kernel.

Ordering principles. In Section 5 we will give different formulations of the total ordering principle. We will see that the propositional translations of the first-order formulation given in [16] have easy refutations (as observed in [16]) because of the presence of a kernel. We emphasize that the same ordering principle requires exponential-size tree-like Resolution refutations in the non-parameterized setting [12].

Bounded-width CNF. The kernels in the previous examples are very explicit, but this is not always the case (as in vertex cover, see Section 1.3.2). Is it easy to find a kernel if it is known to exist? The answer to this question has consequences regarding automatizability of tree-like Parameterized Resolution. We see now a general strategy for finding kernels and fpt-bounded refutations for parameterized contradictions of bounded width. This technique essentially generalizes the one used for vertex cover explained in the introduction.

Theorem 11. *If F is a CNF of width d and (F,k) is a parameterized contradiction, then (F,k) has a tree-like Parameterized Resolution refutation of size $O(d^{k+1})$. Moreover, there is an algorithm that for any (F,k) either finds such tree-like refutation or finds a satisfying assignment for F of weight $\leq k$. The algorithm runs in time $O(|F| \cdot k \cdot d^{k+1})$.*

Proof. Assume (F,k) is a parameterized contradiction. We want to find a refutation for F with parameter k (i. e., at most k variables can be set to true). We first consider a clause $C = x_1 \vee x_2 \vee \ldots \vee x_l$ where $l \leq d$ with all positive literals. Such clause exists because otherwise the full zero assignment would satisfy F.

By induction on k we will prove that (F,k) has a parameterized tree-like refutation of size at most $2 \cdot \sum_{i=0}^{k+1} d^i - 1$. For $k = 0$ the clauses $\{\neg x_i\}_{i=1}^{l}$ are parameterized axioms of the system, thus C is refutable in size at most $1 + 2l \leq 1 + 2d$.

Now consider $k > 0$. For any $1 \leq i \leq l$, let F_i be the restriction of F obtained by setting $x_i = 1$. Each $(F_i, k-1)$ is a parameterized contradiction, otherwise (F,k) would not be. By inductive hypothesis $(F_i, k-1)$ has a tree-like refutation of size at most $s = 2\sum_{i=0}^{k} d^i - 1$. This refutation can be turned into a tree-like derivation of $\neg x_i$ from (F,k). Now we can derive all $\neg x_i$ for $1 \leq i \leq l$ and refute clause C. Such refutation has length $1+l+ls \leq 1+d+ds = 2\cdot\sum_{i=0}^{k+1} d^i - 1$.

By inspection of the proof, it is clear that the refutation can be computed by a simple procedure which at each step looks for a clause C with only positive literals, and builds a refutation of (F,k) recursively by: building l refutations of $(F_i, k-1)$; turning them in l derivations

$(F,k) \vdash \neg x_i$; and resolving against C. This procedure can be easily implemented in the claimed running time.

So far we considered (F,k) to be a parameterized contradiction. If that is not the case, then the algorithm fails. It can fail in two ways: (a) it does not find a clause with only positive literals; (b) one among $(F_i, k-1)$ is not a parameterized contradiction. The algorithm will output the full zero assignment in case (a) and $\{x_i = 1\} \cup \alpha$ in case (b), where α is an assignment witnessing $(F_i, k-1) \notin \mathrm{PCon}$. By induction we can show that on input (F,k) this procedure returns a satisfying assignment of weight $\leq k$. □

We state two interesting consequences of this result.

Corollary 12. *For each $d \in \mathbb{N}$, the set of all parameterized contradictions in d-CNF has a kernel.*

Proof. The refutations constructed in Theorem 11 contain $O(d^k)$ initial clauses in $O(d^{k+1})$ variables. These clauses form a kernel. □

The following corollary expresses some restricted form of automatizability (cf. also the discussion in Section 7).

Corollary 13. *If $\Gamma \subseteq \mathrm{PCon}$ has a kernel, then there exists an fpt-algorithm which on input $(F,k) \in \Gamma$ returns both a kernel and a refutation of (F,k).*

Proof. Let Γ have a kernel of size $f(k)$. Then the kernel only contains clauses of width $\leq f(k)$. On input (F,k) we run the algorithm of Theorem 11 on the CNF formula consisting of all clauses of F with width $\leq f(k)$. This yields a kernel together with its refutation. □

Tseitin tautologies. Fix a bipartite graph $G = (L, R, E)$ such that the degree of the vertices on the left side is constant and the degree of all vertices on the right side is even. Fix now an arbitrary boolean function $f : L \to \{0,1\}$ such that $\sum_{u \in L} f(u) \equiv 1 \pmod 2$. The Tseitin tautology for (G,f) claims that there is no way to define $g : R \to \{0,1\}$ such that for any $u \in L$, $\sum_{v \in \Gamma(u)} g(v) \equiv f(u) \pmod 2$. This fact follows by a simple parity argument. The CNF formulation of this claim uses variables x_v for $v \in R$. The CNF is constituted by the encoding of the constraints $\sum_{v \in \Gamma(u)} x_v \equiv f(u) \pmod 2$ for every $u \in L$. Each linear constraint requires exponential size to be represented in CNF, but this is not an issue here because left-side vertices have constant degree. Hence Tseitin formulas have bounded width. By Theorem 11 they have a kernel, but in contrast to our previous examples this kernel is not very explicit.

Corollary 14. *There are formulas (e. g. Tseitin tautologies) which are hard for general Resolution but easy for tree-like Parameterized Resolution.*

5 Ordering Principles

In this section we discuss Parameterized Resolution refutations for various *ordering principles* OP, also called *least element principles*. The principle claims that any finite partially ordered set has a minimal element. There is a direct propositional translation of OP to a family OP_n of CNF's. Each CNF OP_n expresses that there exists a partially ordered set of size n such that any element has a predecessor. We are also interested in the *linear ordering principle* LOP in which the set is required to be *totally* ordered.

Dantchev, Martin, and Szeider [16] show that a propositional formulation of LOP has small refutations in tree-like Parameterized Resolution. They also show that such efficient refutation does not exists for OP. We observe that their formulation of LOP has short proofs because

it contains very simple kernels. We describe LOP^*, an alternative formulation of the linear ordering principle which does not contain a kernel but nevertheless has (less trivial) fpt-bounded tree-like refutations.

We now describe the three propositional formulations of the ordering principles. For a model with n elements, OP_n, LOP_n, and LOP^*_n are three CNF's over variables $x_{i,j}$ for $i \neq j$ and $i, j \in [n]$.

OP: the general ordering principle has the following clauses:

$\neg x_{i,j} \vee \neg x_{j,i}$	for every i, j	(Antisymmetry)
$\neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k}$	for every i, j, k	(Transitivity)
$\bigvee_{j \in [n] \setminus \{i\}} x_{j,i}$	for every i	(Predecessor)

LOP: is the same as OP with the addition of totality constraints:

$x_{i,j} \vee x_{j,i}$	for every i, j	(Totality)

LOP*: is a different encoding of LOP where we consider only variables $x_{i,j}$ for $i < j$. The intended meaning is that $x_{i,j}$ is true whenever j precedes i in the ordering, and false if i precedes j. The reader may think $x_{i,j}$ to indicate if i and j are an inversion in the permutation for the indexes described by the total order. In particular the full true assignment represents the linear order $(n, n-1, n-2, \ldots, 2, 1)$ while the full false assignment represents $(1, 2, \ldots, n-2, n-1, n)$. This representation will help in the proof of Theorem 15.

LOP^*_n is obtained by substituting in LOP_n any occurrence of $x_{j,i}$ for $j > i$ with $\neg x_{i,j}$. In this way all totality and antisymmetry clauses vanish, and transitivity translates according to relative ranks of the involved indexes.

$\neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k}$	for all $i < j < k$	(Transitivity 1)
$x_{ij} \vee x_{j,k} \vee \neg x_{i,k}$	for all $i < j < k$	(Transitivity 2)
$\bigvee_{j<i} \neg x_{j,i} \vee \bigvee_{i<j} x_{i,j}$	for all i	(Predecessor)

Both OP and LOP are the canonical propositional translations of the first-order formulations of the general and total ordering principle, respectively. In [16] the upper bound for LOP and the lower bound for OP are proved by model-theoretic criteria on the first-order logic formulations.

We remark that in the non-parameterized setting, neither OP, LOP, nor LOP^* have short tree-like Resolution refutations [12], but all of them have general Resolution refutations of polynomial size [33]. It is interesting that in the parameterized setting LOP and LOP^* become easy for tree-like, while OP remains hard. Thus, OP provides a separation between tree-like and dag-like Parameterized Resolution.

It is easy to see that LOP has short tree-like refutations in Parameterized Resolution: notice that the totality clauses for any $k+1$ pairs of indexes form a parameterized contradiction of $2k+2$ variables at most, and so they are a kernel. Unfortunately, LOP is easy to refute for uninteresting reasons: the kernel is very simple. The alternative formulation LOP^* does not have a kernel because all clauses of bounded width are satisfiable by the all zero assignment which represents a total order. Nevertheless LOP^* admits fpt-bounded tree-like refutations.

Theorem 15. *The formulas LOP^*_n have fpt-bounded tree-like refutations in Parameterized Resolution.*

Proof. Let (LOP^*_n, k) be the given instance and assume w. l. o. g. that $k \leq n$. We are going to derive LOP^*_{k+1} from LOP^*_n in polynomial length. This concludes the proof of the theorem because LOP^*_{k+1} has $O(k^2)$ variables and consequently has a tree-like refutation of length $2^{O(k^2)}$.

The idea of the refutation is that for any total order either the least element is among $1, \ldots, k+1$ or there is an element less than all of them. This means that there are at least $k+1$ inversions with respect to the canonical order (i. e., $k+1$ variables are set to 1). To obtain LOP^*_{k+1} we have to derive

$$\bigvee_{1 \leq j < i} \neg x_{j,i} \vee \bigvee_{i < j \leq k+1} x_{i,j}$$

for any $1 \leq i \leq k+1$. W. l. o. g. we discuss the case $i = 1$ which requires simpler notation, the other k cases are analogous.

Our goal then is to derive $\bigvee_{1<j\leq k+1} x_{1,j}$. For any $l > k+1$ consider the following clauses: the first is an axiom of Parameterized Resolution, the others are transitivity axioms.

$$\neg x_{1,l} \vee \neg x_{2,l} \vee \ldots \vee \neg x_{k+1,l} \tag{2}$$
$$x_{1,2} \vee x_{2,l} \vee \neg x_{1,l} \tag{3}$$
$$x_{1,3} \vee x_{3,l} \vee \neg x_{1,l} \tag{4}$$
$$\vdots$$
$$x_{1,k+1} \vee x_{k+1,l} \vee \neg x_{1,l} \tag{5}$$

By applying Resolution between clause (2) and the transitivity clauses we obtain

$$x_{1,2} \vee x_{1,3} \vee \ldots \vee x_{1,k+1} \vee \neg x_{1,l} \tag{6}$$

We just proved that if 1 is the least index among the first $k+1$, then no index above $k+1$ can be less than 1, otherwise there would be at least $k+1$ true variables. The predecessor constraint for 1 contains the literal $x_{1,l}$ for every l; thus applying Resolution between that and clause (6) for every $l > k+1$ yields $\bigvee_{1<j\leq k+1} x_{1,j}$.

We obtained the predecessor axiom for index 1 in LOP^*_{k+1} by a derivation of size $O(kn)$. With $k+1$ such deductions we obtain LOP^*_{k+1}. As the whole refutation of LOP^*_n has length $O(k^2 n) + 2^{O(k^2)}$, it is fpt-bounded. □

The following theorem has been first proved in [16]. Their proof is based on a model-theoretic criterion. We give a combinatorial proof based on Prover-Delayer games.

Theorem 16. *Any tree-like Parameterized Resolution refutation of (OP_n, k) has size $n^{\Omega(k)}$.*

Proof. Let α be an assignment to the variables of OP. The Delayer will keep the following information:

- $G(\alpha) = (V(\alpha), E(\alpha))$ the graph obtained taking as edges the (i,j)'s such that $\alpha(x_{i,j}) = 1$;
- $G^*(\alpha)$ the transitive closure of $G(\alpha)$ and $G^T(\alpha)$ the transpose graph of $G(\alpha)$.

In particular, for any vertex j in $G(\alpha)$, the Delayer considers the following information

- $z_j(\alpha) = |\{i \in [n] \mid \alpha(x_{i,j}) \text{ is not assigned}\}|$,
- $Pred_j(\alpha) = \{i \in [n] \mid \alpha(x_{i,j}) = 1\}$, and
- $PPred_j(\alpha)$ the subset of $Pred_j(\alpha)$ of those edges set to 1 by the Prover .

Loosely speaking the Delayer, taking as few decisions as possible, wants to force: (1) the game to end on a parameterized clause, and (2) the Prover to decide only one predecessor for each node. To reach the former, in some cases she will be forced to decide a predecessor of a node j to avoid that after few more trivial queries the game ends on a predecessor clause. To get (2) she will be forced to say that some node can't be predecessor of some node j. In both cases we will prove that Delayer will keep her number of decisions bounded.

Let α be the assignment built so far in the game and let $x_{i,j}$ be the variable queried by Prover. Delayer acts as follows:

1. if $(i,j) \in E(\alpha)^*$, then answer 1;
2. if $(i,j) \in (E(\alpha)^*)^T$, then answer 0;
3. if $|Pred_j(\alpha)| = 0$ and $z_j(\alpha) \leq k+1$, then answer 1;
4. if $|PPred_j(\alpha)| \geq 1$, then answer 0;
5. otherwise, she leaves the decision to the Prover.

To simplify the argument we assume that in the game, after each decision by the Prover or after a decision by the Delayer according to Rule (3), the Prover asks all variables corresponding to edges that are in $G^*(\alpha)$ and $(G(\alpha)^*)^T$ but not in $G(\alpha)$. This will not change our result since on these nodes Delayer does not score any point.

Let $P^\epsilon(t)$ be the set of edges set to $\epsilon \in \{0,1\}$ by the Prover after stage t ends. Let $D^\epsilon(t)$ be the set of edges set to $\epsilon \in \{0,1\}$ by the Delayer. Finally, let $D^*(t) \subseteq D^1(t)$ be the set of edges set to 1 by the Delayer according to Rule (3) of her strategy. $P^\epsilon_j(t)$, $D^\epsilon_j(t)$, and $D^*_j(t)$ are the subsets of the respective sets formed by those edges having end-node j, i. e., edges of the form (i,j) for some i.

Let α_t be the assignment built after stage t and let α^*_t be the extensions of α_t obtained by assigning all edges from $G^*(\alpha_t)$ to 1 and all edges from $(G(\alpha_t)^*)^T$ to 0. We define $N_j(t) = \{\, (i,j) \mid i \in [n], (i,j) \in dom(\alpha^*_t) \setminus P^0(t) \,\}$.

Lemma 17. *At each stage t of the game, it holds:*

1. $|P^1(t)| + |D^*(t)| \geq \sqrt{|E(\alpha_t)|}$;
2. *if* $|P^1_j(t)| + |D^*_j(t)| = 0$, *then* $|N_j(t)| \leq k$;
3. *if* $w(\alpha_t) \leq k$, *then* α^*_t *does not falsify any predecessor clause;*
4. *for each* $j \in [n]$, $|D^*_j(t)| \leq 1$ *and* $|P^1_j(t)| \leq 1$.

Proof. Condition (1) follows since $|P^1(t)| + |D^1(t)| = |E(\alpha_t)|$, and $|E(\alpha_t)| \leq |E^*(\alpha_t)| \leq (|P^1(t)| + |D^*(t)|)^2$.

Condition (2): $|P^1_j(t)| + |D^*_j(t)| = 0$ implies that the vertex j has no predecessor. The only way to set a predecessor to a vertex which already has one is by Rule (1), but a vertex without predecessors cannot get one by transitive closure. Then an edge $x_{i,j}$ is in $dom(\alpha^*_t) \setminus P^0(t)$ if and only if i is a successor of j in $G^*(\alpha_t)$. Hence there must be a directed tree rooted in j and containing all such successors. As $G(\alpha_t)^T$ contains at most k edges, there are at most k successors of j. Hence $|N_j(t)| \leq k$.

Condition (3): consider a predecessor clause C_j which is not satisfied by α_t. Then there are at least $k+1$ variables $x_{i,j}$ unset, since otherwise, according to Rule (3) Delayer should have set one predecessor for j. If $|P^1_j(t)| \geq 1$, then C_j would be satisfied. Then by $|P^1_j(t)| + |D^*_j(t)| = 0$

and by condition (2) at most k additional literals of C_j are set to 0 by α_t^*. The claim follows since there is at least one unset literal in C_j.

Condition (4): the first time that a predecessor of some node j is decided in the game is either by a decision of the Prover or by a decision of the Delayer according to Rule (3). Since Delayer applies Rule (3) only in the case no predecessor has been yet decided, it follows that $|D_j^*(t)| \leq 1$. Moreover, by Rule (4) Delayer prevents the Prover to set more than one predecessor for each node, hence $|P_j^1(t)| \leq 1$. □

Lemma 18. *After the last stage f of the game the following holds:*

- *a parameterized clause is falsified;*
- $|P^1(f)| + |D^*(f)| \geq \sqrt{k+1}$.

Proof. For the first condition, we notice that Rules (1) and (2) in the Delayer's strategy guarantee that neither antisymmetry nor transitivity axioms will be ever falsified during the game. Assuming that α_f has weight strictly less then $k+1$, then by Lemma 17 (part 3), no predecessor clause is falsified. Hence $w(\alpha_f) = k+1$ and a parameterized clause is falsified.

The second property follows by Lemma 17 (part 1) and by $|E(\alpha_f)| \geq w(\alpha_f)$ which is equal to $k+1$ because of the first part of this lemma. □

Set $c_1(x_{i,j}, \alpha) = z_j(\alpha)$ and $c_0(x_{i,j}, \alpha) = \frac{z_j(\alpha)}{z_j(\alpha)-1}$. For a given play of the game, let $t_{i,j}$ be the stage of the game when the variable $x_{i,j}$ is set. Let $sc_j(t)$ be the number of points scored by the Delayer up to stage t for answers of the Prover to the variables $x_{1,j}, x_{2,j}, \ldots, x_{n,j}$. Then the number of points scored by the Delayer at the end of the game is $\sum_{j=1}^{n} sc_j(f)$.

Lemma 19. *The following implications hold*

1. *If $|P_j^1(f)| = 1$, then $sc_j(f) \geq \log n - \log(k+1)$.*
2. *If $|D_j^*(f)| = 1$, then $sc_j(f) \geq \log n - \log(2k+1)$.*

Proof. For the first claim, let $(i,j) \in D_j^*(f)$ and let $t_{i,j}$ be the stage when $x_{i,j}$ was set. We claim that $|P_j^0(t_{i,j})| \geq n - (2k+1)$. W. l. o. g. we can assume that the variables $x_{i',j}$ set to 0 by the Prover are the first ones with end-node j to be set to 0, because $c_0(x_{i',j}, \alpha)$ is strictly decreasing with respect to $z_j(\alpha)$. Hence the Delayer gets at least

$$\sum_{l=n}^{2k+2} \log \frac{l}{l-1} = \log n - \log(2k+1)$$

points on variables $x_{1,j}, \ldots, x_{n,j}$.

It remains to prove the claim that $|P_j^0(t_{i,j})| \geq n - (2k+1)$. According to Rule (3) of the strategy, there are at least $n-(k+1)$ variables $x_{i',j}$ set to 0 in $\alpha_{t_{i,j}}$. Hence $|P_j^0(t_{i,j})| + |D_j^0(t_{i,j})| \geq n - (k+1)$. Since at this stage i is the first predecessor of j to be fixed, then the Delayer has not set variables $x_{i',j}$ to 0 according to Rule (4), but only by Rule (2). Moreover, for the same reason, if t' is the stage preceding $t_{i,j}$ we have that: $|D_j^0(t_{i,j})| = |D_j^0(t')| = |N_j(t')| \leq k$, where the last inequality holds by Lemma 17 (part 2). Then $|P_j^0(t_{i,j})| \geq n - (2k+1)$.

We now show the second claim of the lemma. Let $t_{i,j}$ be the stage in which Prover sets some $x_{i,j}$ to 1, and let α be the partial assignment corresponding to that stage. W. l. o. g. we assume that all variables in $P_j^0(t_{i,j})$ are set before any variable in $D_j^0(t_{i,j})$, because c_0 is monotone decreasing in the size of the second argument. Fix $p = |P_j^0(t_{i,j})|$. By Lemma 17 (part 2) we get

$|N_j(t')| \leq k$ where t' is the stage preceding $t_{i,j}$. Hence we know that $z_j(\alpha) \geq n-k-p$. The amount of points got by Delayer on vertex j is at least

$$\sum_{l=n}^{n-p+1} \log \frac{l}{l-1} + \log(n-k-p) = \log n - \log \frac{n-p}{n-k-p} \geq \log n - \log(k+1) .$$

□

The Delayer scores $\sum_{j=1}^{n} sc_j(f)$. By Lemma 18 there are at least $\sqrt{k+1}$ vertices such that either $|D_j^*(f)| \geq 1$ or $|P_j^1(f)| \geq 1$. For each vertex such events are mutually exclusive by the definition of the rules. Then by Lemma 19 Delayer gets at least $\sqrt{k+1}(\log n - \log(2k+1))$ points. By Theorem 8 we get the lower bound. □

6 Parameterized Resolution is not fpt-bounded

We show a lower bound for the pigeonhole principle in Parameterized Resolution by using the game of Pudlák [27]. While this game is also played between Prover and Delayer, it is very different from our asymmetric Prover-Delayer game of Section 3. In the asymmetric game, Delayer is using *one* strategy which tries to force Prover on a long path, thereby showing that the proof tree is large. In Pudlák's game, the Delayer uses a *family* of strategies (called "superstrategy" by Pudlák) with the aim to force Prover through every part of the proof.

6.1 Parameterized Proofs as Games

We will now explain how Pudlák's game from [27] can be adapted to show lower bounds in Parameterized Resolution. Consider any Parameterized Resolution refutation of a parameterized contradiction (F, k). An arbitrary assignment falsifies the empty clause at the end of the refutation, and because of soundness of the inference rule, one of its predecessor clauses must be falsified as well. The argument can be repeated, finding a falsified predecessor for any falsified clause in the refutation. This defines a walk in the refutation which starts at the empty clause and goes back to one of the initial clauses or a parameterized axiom, touching just falsified clauses. In the case of (Parameterized) Resolution it is easy to determine with a single variable query what predecessor is falsified: if $A \vee B$ is inferred from $A \vee x$ and $B \vee \neg x$, knowing the value of x is sufficient to walk back one step.

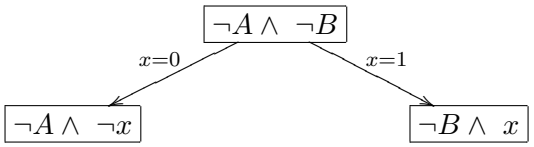
This allows to interpret a refutation as the following interactive process: a *Delayer* claims to know an assignment with at most k variables set to 1 and which satisfies F. The *Prover* queries the Delayer for variable values, can save such values in memory and can forget some of them at will. The Prover wants to expose Delayer lie by exhibiting either an initial or a parameterized clause which is falsified by the assignment. Following Pudlák, we call the memory configuration of the Prover at any step a *record*.

A *Prover strategy* is a finite directed acyclic graph with a single source marked by the empty record. The strategy must (a) define which variable to query in the next step; (b) define what values queried so far are kept in the record; (c) expose the Delayer at any leaf record. The Prover complexity is the size of the strategy, which is the number of distinct records that appear in all possible games (i. e., we ignore the behaviour of Prover on records which are not reachable from the initial empty record).

There is a 1-1 correspondence between Prover strategies and parameterized refutations of (F, k). Namely, a parameterized refutation of (F, k) can be used by Prover to traverse it backwards from the empty clause up to the initial or parameterized clauses, with the invariant

that at each step he keeps in his record the set of values which falsify the clause in the proof. Any inference of the refutation translates to a step in such exploration as

$$\frac{A \vee\ x \qquad B \vee\ \neg x}{A \vee\ B} \qquad \text{corresponds to}$$



where Prover saves new information in his record (the value of x), but also deletes some previous information (corresponding to $\neg A$ or $\neg B$). It is easy to see that also the reverse translation from strategies into proofs is possible: the negation of the records in a strategy translate into clauses, deletions into weakening, and queries into inference steps.

6.2 Delayer Strategies as Refutation Lower Bounds

Because of this correspondence between refutations and Prover strategies, we want to show that Prover always needs a high ($n^{\Omega(k)}$) number of records to expose the Delayer. Notice that a Prover strategy which never forgets information reaches a falsified initial clause in less steps, but then a record occurs in less games. Instead, a record appearing in different games typically contains little information, thus a strategy of small size may require Prover to forget a lot of information. Indeed a Prover strategy which never forgets information corresponds to a tree-like Parameterized Resolution refutation, which is a weaker proof system than general Parameterized Resolution.

So far we argued that Prover strategies represent refutations. We now show that a *randomized Delayer strategy* is the key to obtain lower bounds on the size of such refutations. We may think of such a Delayer strategy as a collection of single strategies (Pudlák [27] calls this a "superstrategy") which force Prover to generate many records. Without loss of generality we also allow Delayer to *give up*, which lets Prover win even before exposing Delayer. The key to lower bounds is then given by the following fact:

Fact 20. *Fix a Prover strategy $\mathcal{P}$ which has a set of records R, and a randomized Delayer strategy $\mathcal{D}$. If for each record $r \in R$*

$$\Pr_{\mathcal{D}}[\textit{ Prover wins on record } r\ ] \leq \frac{1}{S}$$

then $|R| \geq S$.

Proof. The probability that strategy P wins against the Delayer is at most $\frac{|R|}{S}$ by union bound, and is at least 1 by definition. This proves the claim. □

6.3 The Lower Bound for the Pigeonhole Principle

Theorem 21. *Any refutation of (PHP_n^{n+1}, k) in Parameterized Resolution requires size $n^{\Theta(k)}$.*

Proof. For the upper bound we can easily build a decision tree which explores all $n^{O(k)}$ assignments with weight at most k.

The lower bound is based on the game interpretation of a Resolution refutation. Any line in the refutation is a game position between Prover and Delayer. We show that for any refutation (i. e., any Prover) there exists a randomized Delayer strategy such that the probability of a game position to be a winning position for the Prover is very small. The Prover wins with probability 1, so there must be a lot of winning positions (i. e., lines in the refutation).

During the game the Prover record is represented as a matrix P with $n+1$ rows and n columns, with values in $\{0,1,*\}$. An entry $P(i,j)=1$ means that Prover has recorded the information of pigeon i going into hole j, and likewise $P(i,j)=0$ means that pigeon i is not in hole j. For $P(i,j)=*$, Prover does not have any information. Thus the matrix P corresponds to a partial assignment to the variables $x_{i,j}$. The Prover wins whenever this assignment falsifies either an initial clause from PHP_n^{n+1} or a parameterized clause (i. e., there are at least k ones on the record), or if the Delayer surrenders.

We specify the following sets of "forbidden holes" for a pigeon i,

$$F(i) = \left\{ j \in [n] \;\middle|\; \begin{array}{lll} \text{either} & P(i,j)=0 & \\ \text{or} & P(i',j)=1 & \text{for some } i' \neq i \\ \text{or} & P(i,j')=1 & \text{for some } j' \neq j \end{array} \right\}$$

and we say that pigeon i has $|F(i)|$ forbidden holes on record P (or just "on the record" if P is clear from the context).

The Delayer strategy. The Delayer chooses uniformly at random a partial matching α of $n-2k$ pigeons to $n-2k$ holes. In the game, if Prover asks about a pigeon from $dom(\alpha)$, then Delayer will always answer according to the matching α. We denote $\overline{dom}(\alpha) = [n+1] \setminus dom(\alpha)$ and $\overline{rng}(\alpha) = [n] \setminus rng(\alpha)$. These are the unassigned pigeons and holes for which Delayer has not made a decision before the start of the game. During the game Delayer only cares about the cells in P indexed by $\overline{dom}(\alpha) \times \overline{rng}(\alpha)$, because only here Prover can force her into a contradiction with an initial clause. We call P' the minor of P of size $(2k+1) \times (2k)$ on rows from $\overline{dom}(\alpha)$ and columns from $\overline{rng}(\alpha)$. The Delayer keeps also another private matrix D not known to the Prover, also of size $(2k+1) \times (2k)$. This matrix essentially encodes the "next answer" for any Prover question in $\overline{dom}(\alpha) \times \overline{rng}(\alpha)$.

The Delayer strategy is completely specified by how the Delayer answers to the questions and how she computes D at each step. The Delayer answers to the Prover question $x_{i,j}$ according to the following scheme:

1. For pigeons $i \in dom(\alpha)$, Delayer answers 1 if $\alpha(i) = j$ and 0 if $\alpha(i) \neq j$.
2. For pigeons $i \notin dom(\alpha)$ and holes $j \in rng(\alpha)$, Delayer answers 0.
3. For pigeons $i \notin dom(\alpha)$ and holes $j \notin rng(\alpha)$, Delayer answers $D(i,j)$.

The answers are both based on α and D. The matrix D is recomputed at each step, according to the values of P'. Let $G = \{\, i \in \overline{dom}(\alpha) : |F(i) \cap \overline{rng}(\alpha)| \geq k \,\}$, i. e., G contains all pigeons not assigned by α for which Prover has already excluded more than k of the remaining $2k$ holes left free by α. Because Delayer does not want to allow Prover to easily reach a contradiction on the pigeons from G (for instance, Prover could just continue to query holes for some pigeon $i \in G$), Delayer secretly assigns a hole to each pigeon in G. To achieve this, Delayer defines D as an arbitrary completion of P' (i. e., $*$'s are replaced by 0/1), such that D represents a partial matching which assigns exactly all pigeons from G to some set of holes from $\overline{rng}(\alpha)$. More precisely, D is *any* matrix in $\{0,1\}^{(2k+1)\times(2k)}$ which satisfies:

1. D is a completion of P', i. e., $D(i,j) = P'(i,j)$ for $P'(i,j) \neq *$.
2. Exactly the pigeons in G are matched, i. e.,
   (a) For any $i \in G$ there exists $j \in \overline{rng}(\alpha)$ such that $D(i,j) = 1$;
   (b) For any $i \in \overline{dom}(\alpha) \setminus G$ and any $j \in \overline{rng}(\alpha)$, $D(i,j) = 0$.

3. D is a matching, i. e.,

   (a) For any $i, i' \in \overline{dom}(\alpha)$, $i \neq i'$, and $j \in \overline{rng}(\alpha)$, either $D(i,j) = 0$ or $D(i',j) = 0$;

   (b) For any $i \in \overline{dom}(\alpha)$ and $j, j' \in \overline{rng}(\alpha)$, $j \neq j'$, either $D(i,j) = 0$ or $D(i,j') = 0$.

It is possible that such D does not exists: in this case *the Delayer surrenders*. Given the choice of α, the strategy of the Delayer is completely determined. Thus we will also call α the strategy of the Delayer.

Losing positions for the Delayer. The idea of the Delayer strategy is to play honestly on a large fraction of the variables, while trying to cheat only on the cells of P'. The Delayer always answers consistently with a partial matching, so the only way for Prover to expose her is to either find k assigned ones, or to force her to give up. We characterize the particular kinds of records where the game may end.

Claim 22. *If the Prover wins, at that time in the game either P contains at least k ones, or G contains at least k pigeons.*

Proof. We first see that Prover never wins by falsifying a clause from PHP_n^{n+1}. By definition of her strategy, Delayer always answers according to a partial matching which matches all pigeons from $dom(\alpha) \cup G$. Thus no conflict between pigeons arises. Also P can never falsify any of the big initial clauses $\bigvee_{j \in [n]} x_{i,j}$. This is clear for pigeons $i \in dom(\alpha)$ because α assigns some hole to i. But it also holds for pigeons $i \in \overline{dom}(\alpha)$ because after there are k forbidden holes for i from $\overline{rng}(\alpha)$, Delayer secretly reserves a hole for i in D. Thus, during the game there can never be more than $n - k$ forbidden holes for some pigeon without Delayer having already reserved some hole for it.

It remains to show that if $|G| \leq k$, then Delayer can always find a D as required by the strategy, thus she does not surrender. Consider the private matrix D_{old} computed by the Delayer at the previous step and denote G_{old} to be set of pigeons matched in D_{old}. If in the previous step of the game Prover only deleted some information from the record, then $G \subseteq G_{old}$ and thus Delayer can compute D. Assume now that Prover increased the information on the record and therefore $G \supseteq G_{old}$. The elements in $G \setminus G_{old}$ have exactly k forbidden holes because in each step of the game the number of forbidden holes per pigeon can increase at most by one, and they where below the threshold k at the previous step. Consider the matching induced by D_{old}. Each element of $G \setminus G_{old}$ still has k available holes, thus even excluding the holes occupied by pigeons from G_{old}, they still have at least $k - |G_{old}|$ available holes each. A simple greedy matching strategy will be sufficient to match all pigeons from $G \setminus G_{old}$ of which there are $\leq k - |G_{old}|$ many.

We conclude that the game ends either because P contains at least k ones, or because the Delayer surrenders, which means $|G| > k$. □

Intuition for the remaining part of the proof. The proof is a probabilistic argument. For a random α, the probability that a clause in the refutation corresponds to the final position in the interactive game is $\frac{1}{n^{\Omega(k)}}$. Any game ends somewhere, so there must be $n^{\Omega(k)}$ clauses.

The probability estimation is based on two observations: (a) if a game interaction produces a record, this record is compatible with the α chosen by the Delayer; (b) if Delayer gives up, it means that P' contains too much information for the Delayer to continue cheating.

The record analysis is divided in three cases: in cases (I) and (II) we argue that if a record contains a large amount of information, then there is little chance for a random α to be compatible with said record. In particular, case (I) corresponds to the Prover exposing k ones in

the Delayer answers, and in case (II) Prover's record contains a huge amount of zeros. In both cases, Delayer succeeded in forcing Prover to write down a good part from the matching α.

For case (III) we argue about a record with less than k ones and small amount of information. Such record must force the Delayer to give up. This means that the small amount of information must be concentrated on P'. For random α and random relative position of P' with respect to P, this happens with small probability. Intuitively, in case (III) Prover is forced to "guess" at least $\overline{dom}(\alpha)$ or $\overline{rng}(\alpha)$ which he can achieve only with small probability.

Prover probability of winning. We now show that in each game position (i. e., in each line of the refutation), Prover has winning probability $\frac{1}{n^{\Omega(k)}}$ against Delayer's randomized strategy. This proves that any refutation contains $n^{\Omega(k)}$ lines.

Observe that as a necessary condition for a record to be a winning position for Prover, the record has to be compatible with the choice of α. This is because Delayer always answers according to α on pigeons from $dom(\alpha)$, and thus there is no way that information which is inconsistent with α can be written in Prover's record. We will use this fact in the following steps.

We will consider three cases for a record: (I) the record contains exactly k ones, (II) the record contains at least $n^{2/3}+2k+1$ pigeons which each have at least $kn^{1/3}\log n$ forbidden holes, (III) the record has $< k$ ones and contains at most $n^{2/3}+2k$ pigeons which have more than $kn^{1/3}\log n$ forbidden holes each. Those three cases cover all possible records in the game. The result follows by proving that in each case, the probability that a record is a winning position for Prover is $\frac{1}{n^{\Omega(k)}}$.

Case I. Prover has exactly k ones on the record. On such records, the Prover always wins, but only for those game plays that actually lead to Prover's record. A necessary condition for this is that P is compatible with α. Let S be the set of pigeons with a 1 on the record. Then

$$\Pr_\alpha[\,P \text{ is winning against } \alpha\,] \;\le\; \Pr_\alpha[\,P \text{ is compatible with } \alpha\,] \;\le$$
$$\Pr_\alpha\left[\,|S\cap dom(\alpha)| < \frac{k}{2}\,\right] + \Pr_\alpha\left[\,P \text{ is compatible with } \alpha \;\middle|\; |S\cap dom(\alpha)| \ge \frac{k}{2}\,\right] \;.$$

The probability that $|S\cap dom(\alpha)| < \frac{k}{2}$ is equal to the probability that the uniform random set $\overline{dom}(\alpha)\subset[n+1]$ of size $2k+1$ intersects the set $S\subset[n+1]$ of size k in at least $k/2$ positions. Let $\overline{dom}(\alpha)=\{i_1,\ldots,i_{2k+1}\}$. Then

$$\Pr_\alpha\left[\,|S\cap\overline{dom}(\alpha)|\ge\frac{k}{2}\,\right] = \Pr_{i_1,\ldots,i_{2k+1}}\left[\,\exists\, 1\le l_1<\ldots<l_{k/2}\le 2k+1 \text{ s.t. } \{i_{l_1},\ldots,i_{l_{k/2}}\}\subseteq S\,\right] \le$$
$$\le \sum_{1\le l_1<\ldots<l_{k/2}\le 2k+1} \Pr_{i_1,\ldots,i_{k/2}}\left[\,\{i_{l_1},\ldots,i_{l_{k/2}}\}\subseteq S\,\right] \;\le\; \binom{2k+1}{k/2}\cdot\frac{\binom{k}{k/2}}{\binom{n+1}{k/2}} \;\le\; \frac{1}{n^{\Omega(k)}} \;.$$

The probability of α to be compatible with P, given that $|S\cap dom(\alpha)|\ge\frac{k}{2}$, is bounded by the probability of guessing the matching in P for the pigeons in $S\cap dom(\alpha)$, that is

$$\frac{1}{n}\cdot\frac{1}{n-1}\cdot\cdots\cdot\frac{1}{n-|S\cap dom(\alpha)|+1} \;\le\; \frac{1}{n^{\Omega(k)}} \;.$$

This completes Case I.

Case II. The Prover has a set S of at least $n^{2/3}+2k+1$ pigeons with $kn^{1/3}\log n$ forbidden holes each on the record P. At least $n^{2/3}$ of them are in $dom(\alpha)$. As in Case I the probability

of strategy α to lose in position P is bounded by the probability of α being compatible with P. For compatibility it is necessary that a random α does not match any of the (at least) $n^{2/3}$ pigeons in $dom(\alpha)$ with any of their corresponding forbidden holes. Let us consider the matching holes for such pigeons as a randomly chosen sequence $h_1 \dots h_{n^{2/3}}$. Assuming that $h_1 \dots h_{l-1}$ are compatible choices there are at most $n - kn^{1/3}\log n - l + 1$ good choices for h_l over $n - l + 1$ possible choices. Thus h_l is compatible with probability at most

$$\frac{n - kn^{1/3}\log n - l + 1}{n - l + 1} \;=\; 1 - \frac{kn^{1/3}\log n}{n - l + 1} \;\le\; 1 - \frac{k \log n}{n^{2/3}} \;.$$

The probability of the whole sequence to be compatible is then bounded by

$$\left(1 - \frac{k \log n}{n^{2/3}}\right)^{n^{2/3}} \;\le\; \frac{1}{e^{\Omega(k \log n)}} \;\le\; \frac{1}{n^{\Omega(k)}} \;.$$

Case III. Record P does not contain k ones, and the set S of pigeons with more than $kn^{1/3}\log n$ forbidden holes has size at most $n^{2/3} + 2k$. If Prover wins with record P against strategy α, then Claim 22 implies $|G| > k$. Thus the probability of strategy α to lose on record P is at most the probability that in $\overline{dom}(\alpha)$ there are k pigeons with at least k forbidden holes contained in $\overline{rng}(\alpha)$ each. We split the analysis into two further sub-cases.

Case III. (a) Assume first that $|S \cap \overline{dom}(\alpha)| \ge k$. Intuitively, in this case Prover has managed to place a good number of pigeons ($\ge k$) with many forbidden holes into $\overline{dom}(\alpha)$. For these pigeons from $S \cap \overline{dom}(\alpha)$, it will be easy for Prover to exclude $\ge k$ holes of $\overline{rng}(\alpha)$, and thus force these pigeons into G. But in total, S only contains few pigeons ($\le n^{2/3} + 2k$), and this means that Prover has to "guess" $\overline{dom}(\alpha)$ which is hard for him.

For the formal analysis, let $i_1, \dots, i_{2k+1}$ denote the elements of $\overline{dom}(\alpha)$. They form a uniformly chosen set of size $2k + 1$ in $[n + 1]$. The probability that they intersect S in k positions is at most

$$\Pr_{i_1,\dots,i_{2k+1}}[\,|S \cap \{i_1, \dots, i_{2k+1}\}| \ge k\,] \;\le$$

$$\sum_{1 \le l_1 < l_2 < \dots < l_k \le 2k+1} \Pr_{i_{l_1},\dots,i_{l_k}}[\,\{i_{l_1}, \dots, i_{l_k}\} \subseteq S\,] \;\le\; \binom{2k+1}{k} \frac{\binom{n^{2/3}+2k}{k}}{\binom{n}{k}} \;\le\; \frac{1}{n^{\Omega(k)}} \;.$$

Case III. (b) The other possibility is that $|S \cap \overline{dom}(\alpha)| < k$. But then for P to be winning against α, there must exist at least one pigeon $i \in \overline{dom}(\alpha) \setminus S$ such that $|F(i) \cap \overline{rng}(\alpha)| \ge k$. By union bound this is at most $2k+1$ times the probability of such event for a fixed $i \in \overline{dom}(\alpha) \setminus S$. By $j_1, \dots, j_{2k}$ we denote the elements of $\overline{rng}(\alpha)$, they form a uniformly chosen set of size $2k$ in $[n]$. Then

$$\Pr_{j_1,\dots,j_{2k}}[\,|F(i) \cap \{j_1, \dots, j_{2k}\}| \ge k\,] \;\le$$

$$\sum_{1 \le l_1 < l_2 < \dots < l_k \le 2k} \Pr_{j_{l_1},\dots,j_{l_k}}[\,\{j_{l_1}, \dots, j_{l_k}\} \subseteq F(i)\,] \;\le\; \binom{2k}{k} \frac{\binom{kn^{1/3}\log n}{k}}{\binom{n}{k}} \;\le\; \frac{1}{n^{\Omega(k)}} \;.$$

This completes the proof. □

7 Future Perspectives

Alekhnovich and Razborov [2] were the first to establish very interesting connections between (classical) proof complexity and parameterized complexity.

Definition 23. *A proof system P is* automatizable *if there exists an algorithm which for a tautology F with a P-proof of size S finds a P-proof for F of size at most $S^{O(1)}$ and runs in time $S^{O(1)}$.*

Alekhnovich and Razborov [2] proved that if (classical) Resolution was automatizable, then W[P] coincides with FPR, the randomized version of FPT. Since separating FPT from W[2] is a similar but probably harder problem than proving W[P] $\neq$ FPR, proving lower bounds to successively stronger parameterized proof systems on one side, and strengthening the result of Alekhnovich and Razborov on the other seems, in our view, a promising approach towards *unconditional non-automatizability* of (classical) Resolution and other proof systems, which is an important problem in proof complexity.

DPLL is the core of the algorithm to prove that tree-like Resolution is quasi-polynomially automatizable (see [4]). In the analysis of that algorithm the main feature is the observation that in a binary tree of size S one of the two sub-trees is of size at most $S/2$. This suggests that our asymmetric Prover-Delayer game, which provides a finer estimation of the size of the sub-trees, might lead to some new family of algorithms based on DPLL.

The concept of automatizability can be easily extended to parameterized proof systems and it appears to be an interesting problem to investigate. However, there are some differences with respect to (classical) Resolution. Namely, a quasi-polynomial approximation of the shortest proof is meaningless in the context of Parameterized Resolution, because every $(F, k) \in$ PCon with $|F| = n$ has a refutation of size $c \cdot \binom{n}{k+1}$ for some constant c. If $k \leq \log n$ this is smaller than $n^{\log n}$; otherwise $\binom{n}{k+1} \leq 2^{(k+1)^2}$ which is fpt with respect to k. Hence for any $(F, k) \in$ PCon there exists a refutation of size $f(k)q(n)$ where f is some computable function and q is some quasi-polynomial function. We are in the situation discussed in [2], where we want to discriminate between polynomial and quasi-polynomial efficiency. Another difficulty is that known non-automatizability results in [1, 2, 4] all use formulas which are interesting and meaningful on assignments of big weight.

References

[1] M. Alekhnovich, S. R. Buss, S. Moran, and T. Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *J. Symb. Log.*, 66(1):171–191, 2001.

[2] M. Alekhnovich and A. A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, 2008.

[3] P. Beame, J. C. Culberson, D. G. Mitchell, and C. Moore. The resolution complexity of random graph k-colorability. *Discrete Applied Mathematics*, 153(1-3):25–47, 2005.

[4] P. Beame, R. M. Karp, T. Pitassi, and M. E. Saks. The efficiency of resolution and Davis–Putnam procedures. *SIAM J. Comput.*, 31(4):1048–1075, 2002.

[5] P. Beame, H. A. Kautz, and A. Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.*, 22:319–351, 2004.

[6] P. Beame and T. Pitassi. Simplified and improved resolution lower bounds. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science*, pages 274–282, 1996.

[7] P. W. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Mathematical Society*, 73(3):1–26, 1996.

[8] E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

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

[10] A. Blake. *Canonical expressions in boolean algebra.* PhD thesis, University of Chicago, 1937.

[11] M. L. Bonet, J. L. Esteban, N. Galesi, and J. Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000.

[12] M. L. Bonet and N. Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001.

[13] Y. Chen and J. Flum. The parameterized complexity of maximality and minimality problems. *Annals of Pure and Applied Logic*, 151(1):22–61, 2008.

[14] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988.

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

[16] S. S. Dantchev, B. Martin, and S. Szeider. Parameterized proof complexity. In *Proc. 48th IEEE Symposium on the Foundations of Computer Science*, pages 150–160, 2007.

[17] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[18] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[19] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, Berlin Heidelberg, 1999.

[20] J. L. Esteban, N. Galesi, and J. Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321(2–3):347–370, 2004.

[21] J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[22] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, Berlin Heidelberg, 2006.

[23] Y. Gao. Data reductions, fixed parameter tractability, and random weighted d-CNF satisfiability. *Artificial Intelligence*, 173(14):1343–1366, 2009.

[24] A. Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

[25] J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.

[26] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[27] P. Pudlák. Proofs as games. *American Math. Monthly*, pages 541–550, 2000.

[28] P. Pudlák and R. Impagliazzo. A lower bound for DLL algorithms for SAT. In *Proc. 11th Symposium on Discrete Algorithms*, pages 128–136, 2000.

[29] S. Riis. A complexity gap for tree resolution. *Computational Complexity*, 10(3):179–209, 2001.

[30] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

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

[32] N. Segerlind, S. R. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM J. Comput.*, 33(5):1171–1200, 2004.

[33] G. Stalmark. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996.

[34] A. Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987.

Appendix

A Weighted Satisfiability and Completeness for W[1] and W[2]

Deciding whether a parameterized CNF is satisfiable or not is related with complexity classes W[1] and W[2] (see [19, 22, 26] for more information about these classes). In particular, we are interested in the following weighted satisfiability problem for bounded-width formulas:

Problem:	WEIGHTED d-CNF SAT (complete for W[1], see [19])
Input:	F, a CNF where all clauses have at most d literals.
Parameter:	k, a non-negative integer.
Question:	Does there exist a satisfying assignment for F with weight *equal* to k?

The unbounded width version of this problem is defined as:

Problem:	WEIGHTED CNF SAT (complete for W[2], see [19])
Input:	F, a CNF.
Parameter:	k, a non-negative integer.
Question:	Does there exist a satisfying assignment for F with weight *equal* to k?

In Theorem 11 we show that for CNF of constant width the problem of finding an assignment with weight *at most* k is in FPT. Thus W[1]-completeness really depends on the equality requirement (this observation was also made in [13,19][1]). Dantchev et al. [16] claim without full proof that this is not the case for W[2] completeness. We include the full proof for completeness.

Proposition 24 (Dantchev, Martin, Szeider [16])**.** *Deciding if an arbitrary CNF F has a satisfying assignment of weight at most k is* W[2]*-complete.*

Proof. We refer to this problem as the "relaxed" version of Weighted CNF Sat, which is in contrast referred as the "exact" version. Consider $k+1$ new variables $y_1, \ldots, y_{k+1}$. The formula $F \wedge (y_1 \vee y_2 \vee \cdots \vee y_{k+1})$ has a satisfying assignment of weight $k+1$ if and only if F has a satisfying assignment of weight at most k. Furthermore, translating back such assignment to an assignment for F is a simple projection. This proves that the relaxed version is reducible to the exact one, and thus the former is in W[2].

To prove completeness consider the following reduction from the exact version to the relaxed one. Let (F, k) be the input, where F is a CNF in variables $x_1, \ldots, x_n$. Consider the following CNF ψ which in addition to $x_1, \ldots, x_n$ uses new variables $y_{i,j}$ for $i \in [n]$ and $j \in [k]$:

$\bigvee_i y_{i,j}$	for any $j \in [k]$
$\neg y_{i,j} \vee \neg y_{i',j}$	for any $i \neq i' \in [n]$ and $j \in [k]$
$\neg y_{i,j} \vee \neg y_{i,j'}$	for any $i \in [n]$ and $j \neq j' \in [k]$
$y_{i,1} \vee y_{i,2} \vee \ldots \vee y_{i,k} \vee \neg x_i$	for any $i \in [n]$
$\neg y_{i,j} \vee x_i$	for any $i \in [n]$ and $j \in [k]$

Formula ψ is satisfiable if and only if there is a set of k indices in $[n]$ matched with $[k]$. Variable x_i is true if and only if i is in such set. Thus any satisfying assignment for ψ has weight $2k$. The reduction from the exact version to the relaxed version of Weighted CNF Sat is given by $(F, k) \mapsto (F \wedge \psi, 2k)$. This proves that the relaxed version is W[2]-hard. □

Observe that using unbounded clauses is crucial in the previous proof, and that this reduction does not work with bounded width CNF's. In general, any reduction from Weighted d-CNF Sat to its relaxed version would imply $\mathsf{W}[1] \subseteq \mathsf{FPT}$ and $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{o(n)})$ (for the latter implication see [19, Corollary 17.7]).

[1] We thank Martin Grohe for pointing out the reference [13] to us.