The Strength of Parameterized Tree-like Resolution

Olaf Beyersdorff*
Institut für Informatik, Humboldt-Universität zu Berlin, Germany
beyersdo@informatik.hu-berlin.de

Nicola Galesi Massimo Lauria
Dipartimento di Informatica, Sapienza - Università di Roma, Italy
{galesi,lauria}@di.uniroma1.it

Abstract

We examine the proof-theoretic strength of parameterized tree-like resolution—a proof system for the coW[2]*-complete set of parameterized tautologies. Parameterized resolution and, moreover, a general framework for parameterized proof complexity was introduced by Dantchev, Martin, and Szeider (FOCS'07). In that paper, Dantchev et al. show a complexity gap in parameterized tree-like resolution for propositional formulas arising from translations of first-order principles.*

Here we pursue a purely combinatorial approach to obtain lower bounds to the proof size in parameterized tree-like resolution. For this we devise a prover-delayer game suitable for parameterized resolution. By exhibiting good delayer strategies we then show lower bounds for the pigeonhole principle as well as the order principle. On the other hand, we demonstrate that parameterized tree-like resolution is a very powerful system, as it allows short refutations of all parameterized contradictions given as bounded-width CNF's. Thus, a number of principles such as Tseitin tautologies, pebbling contradictions, or random 3-CNF's which serve as hard examples for classical resolution become easy in the parameterized setting.

1 Introduction

Proof complexity is a research field which owes one of its main motivations from the problem of separating complexity classes as P, NP, and coNP, using an approach which integrates techniques and results from mathematical logic, model theory, combinatorics, and computational complexity. Cook and Reckhow [11] initiated the study of lengths of proofs in propositional proof systems. Their result that the existence of a *polynomially bounded* propositional proof system, i. e., a proof system where all tautologies have polynomial-size proofs, is equivalent to NP = coNP, has opened the way to proving lower bounds for the lengths of proofs in a diversity of propositional proof systems ranging from restricted versions of resolution to bounded-depth Frege systems (see [19] for a recent survey on the field). While all these systems are known to be *not* polynomially bounded, still a lot of effort has to be invested to reach, for instance, super-polynomial lower bounds for Frege systems.

Parameterized complexity is a branch of complexity theory where problems are analyzed in a finer way than in the classical approach: instead of expressing the complexity of a problem as a function only of the input size, one parameter is part of the input instance, and one investigates the effect of the parameter on the complexity. We say that a problem is *fixed-parameter tractable* 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 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 W[P] (see [15,16] for a complete treatment of the field).

Recently, Dantchev, Martin, and Szeider [12] introduced and 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 main motivation behind their work 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.

Tree-like resolution is a well-known propositional proof

*This work was done while the first author was visiting Sapienza University of Rome. Research was supported by DFG grant KO 1053/5-2.

system where proofs are in form of trees. It is one of the most intensively investigated proof systems both from a proof-theoretic point of view ([4, 7–9] among many others) and from an applied perspective, given its importance in the study of algorithms for the satisfiability problem [5, 20]. Dantchev et al. [12] study the complexity of parameterized proofs starting from the parameterized version of tree-like resolution. They prove an extension of Riis' *gap theorem* for tree-like resolution [18], getting a separation for the complexity of 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.

In this work we continue the study of the complexity of proofs in parameterized tree-like resolution. As our main contribution (Section 3) we devise a purely combinatorial approach, based on a prover-delayer game, to characterize proof size in parameterized tree-like resolution. In particular, we use our characterization to prove lower bounds. Our game is inspired by the prover-delayer game of Pudlák and Impagliazzo [17], which is one of the canonical tools to study lower bounds in tree-like resolution [7, 17] and tree-like $Res(k)$ [13]. In fact, we show that the game from [17] is a very simple case of our game.

Using this game, lower bounds to the proof size in parameterized tree-like resolution immediately follow from good strategies for the delayer. We provide such strategies for the case of the pigeonhole principle (Theorem 3.2) and for the case of a (partial) ordering principle (Theorem 5.2). Moreover, our game makes explicit a connection between *complexity* and *information content* of parameterized tree-like resolution proofs, which may be of independent interest and suggest new connections to study the length of proofs.

But what is the real proof strength of parameterized tree-like resolution? Is it similar to that of tree-like resolution? Dantchev et al. indicated that this is not the case. They proved that a propositional encoding of the first-order formulation of the linear ordering principle has short tree-like proofs in parameterized resolution, whereas this principle is hard for usual tree-like resolution [10].

We further investigate classes of parameterized contradictions that are efficiently provable in tree-like resolution, obtaining surprising results. The notion of *efficient kernelization* (see [15]) of languages 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 proved in parameterized tree-like resolution. As an immediate consequence several examples of formulas hard for tree-like resolution are instead efficiently provable 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 4.2 we prove that contradictions of bounded width have a kernel and thus very efficient tree-like resolution proofs. This implies that formulas known to be hard for systems even stronger than (dag-like) resolution like Tseitin tautologies and random 3-CNF's are efficiently provable in parameterized tree-like resolution.

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 prover-delayer game and establish its precise relation to proof size in parameterized tree-like resolution. We also show the 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 (Theorem 5.2). Finally, in Section 6 we introduce the concept of automatizability in the context of parameterized proof systems and discuss several open problems. Due to space limitations some proofs are moved to the appendix.

2 Preliminaries

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 satisfiability* containing instances (F, k) with a propositional formula F in CNF and a parameter $k \in \mathbb{N}$. Weighted CNF satisfiability 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 W[2]-complete problem (cf. [12]). Like in the classical duality between tautologies and satisfiability, we obtain a complete problem for coW[2]:

Definition 2.1 (Dantchev, Martin, Szeider [12])**.** *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 [15, 16].

2.2 Parameterized Proof Systems

We start with the general definition of a parameterized proof system of Dantchev, Martin, and Szeider [12].

Definition 2.2 (Dantchev, Martin, Szeider [12])**.** *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 ftp-time in the proof parameter. We contrast this with the classical concept of Cook and Reckhow where proofs are verified in polynomial time:

Definition 2.3 (Cook, Reckhow [11])**.** *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 2.2, 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 2.2 is that proofs are now verified in polynomial time. We will argue in Theorem 2.7 that this weaker and simpler notion is in fact equivalent to the notion of parameterized proof systems from Definition 2.2 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 2.4 (Dantchev, Martin, Szeider [12])**.** *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 2.5. *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 languages admit ftp-bounded proof systems. Recall that by the theorem of Cook and Reckhow [11], 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 2.6 (Flum, Grohe [14])**.** *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 [11] 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 2.7. *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.*

As in items 1 and 2 of Theorem 2.7, 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. 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. Undoubtedly, resolution is the most studied and best-understood propositional proof system (cf. [19]).

For the remaining part of this paper we will concentrate on *parameterized resolution* as introduced by Dantchev, Martin, and Szeider [12]. Parameterized resolution is a refutation system for the set PCon of parameterized contradictions (cf. Definition 2.1). 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 *parameterized tree-like resolution*. As explained in [12], a parameterized tree-like 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 parameterized tree-like 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 Lower Bounds via a Prover-Delayer Game

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 parameterized tree-like resolution is given by the next theorem:

Theorem 3.1. *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 parameterized tree-like 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 provers choice is $x = j$ with $j \in \{0, 1\}$, then we get

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

This completes the proof of the induction. □

Notice that for 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 [17]. 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

$$\bigvee_{j\in[n]} x_{i,j} \quad \text{for all pigeons } i \in [n + 1] \text{ and } \quad \neg x_{i_1,j} \vee \neg x_{i_2,j}$$

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 parameterized tree-like resolution.

Theorem 3.2. *PHP_n^{n+1} has no fpt-size parameterized tree-like resolution refutation.*

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 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 a hole yet, there are at least k holes where it could go (and of these only $w(\alpha)$ holes are 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

$$\begin{aligned} \sum_{z=0}^{n-2k-1} \log \frac{n-z}{n-z-1} &= \log \prod_{z=0}^{n-2k-1} \frac{n-z}{n-z-1} \\ &= \log \frac{n}{2k} = \log n - \log 2k \end{aligned}$$

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 3.1, we obtain $(\frac{n}{2k})^k$ as a lower bound to the size of each parameterized tree-like resolution refutation of (PHP_n^{n+1}, k). □

By inspection of the above delayer strategy it becomes clear that the lower bound from Theorem 3.2 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. [15]). For parameterized proof complexity we suggest a similar notion of a kernel:

Definition 4.1. *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 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 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 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 [7].

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, V of $n+1$ pigeons and n 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 these principles are hard for general resolution [8]. $PHP(G)$ 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 [12] have easy refutations (as observed in [12]) 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 [10].

Bounded-width CNF. The kernels in the previous examples are very explicit, but this is not always the case. 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.

Theorem 4.2. *If* F *is a CNF of width* w *and* (F, k) *is a parameterized contradiction, then* (F, k) *has a parameterized tree-like resolution refutation of size* $O(w^{k+1})$. *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* $O(|F| \cdot k \cdot w^{k+1})$.

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 $l \leq w$ 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 $2 \cdot \sum_{i=0}^{k+1} w^i - 1$. 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 $1 + 2l \leq 1 + 2w$.

Let now $k > 0$. 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 $s = 2\sum_{i=0}^{k} w^i - 1$. 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 $1 + l + ls \leq 1 + w + ws = 2 \cdot \sum_{i=0}^{k+1} w^i - 1$.

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

Proof. The refutations constructed in Theorem 4.2 contain $O(w^k)$ initial clauses in $O(w^{k+1})$ variables. These clauses form a kernel. □

The following corollary expresses some restricted form of automatizability (cf. also the discussion in Section 6).

Corollary 4.4. *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 4.2 on 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 4.2 they have a kernel, but in contrast to our previous examples this kernel is not very explicit.

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 [12] show that a propositional formulation of LOP has small refutations in parameterized tree-like 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:

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

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 5.1.

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.

$$\begin{array}{lll}
\neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k} & \text{for all } i<j<k & \text{(Transitivity 1)} \\
x_{ij} \vee x_{j,k} \vee \neg x_{i,k} & \text{for all } i<j<k & \text{(Transitivity 2)} \\
\bigvee_{j<i} \neg x_{j,i} \vee \bigvee_{i<j} x_{i,j} & \text{for all } i & \text{(Predecessor)}
\end{array}$$

Both OP and LOP are the canonical propositional translations of the first-order formulations of the general and total ordering principle, respectively. In [12] 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 [10], but all of them have general resolution refutations of polynomial size [21]. 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 5.1. *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^{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 target 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.

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

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 $O(k^2 n) + O(2^{k^2})$, it is fpt-bounded. □

The following theorem has been first proved in [12]. Their proof is based on a model-theoretic criterion. We give a combinatorial proof based on prover-delayer games.

Theorem 5.2. *OP has no fpt-bounded tree-like parameterized resolution refutations.*

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 of the prover or after a decision of the delayer by 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 5.3. *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 $|P^1(t)| \geq 1$, then C_j would be satisfied. Then by $|P^1(t)| + |D^*(t)| = 0$ and by condition (2) at most k additional literals of C_j are set to 0 by α^*_t. The claim follows.

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 $|P^1_j(t)| \leq 1$. □

Lemma 5.4. *After the last stage f of the game the following holds:*

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

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 5.5.

1. *If* $|P^1_j(f)| = 1$, *then* $sc_j(f) \geq \log n - \log(k+1)$.
2. *If* $|D^*_j(f)| = 1$, *then* $sc_j(f) \geq \log n - \log(2k+1)$. □

The delayer scores $\sum_{j=1}^n sc_j(f)$. By Lemma 5.4 there are at least $\sqrt{k+1}$ vertices such that either $|D^*_j(f)| \geq 1$ or $|P^1_j(f)| \geq 1$. For each vertex such events are mutually exclusive by the definition of the rules. Then by Lemma 5.5 delayer gets at least $\sqrt{k+1}(\log n - \log(2k+1))$ points. By Theorem 3.1 we get the lower bound. □

6 Open problems

This paper leaves some questions open to further inquiry. We discuss two of them we consider worth the attention of the community.

Automatizability of parameterized resolution. It is known that in the non-parameterized setting neither tree-like nor general resolution seems to be automatizable [2] and that tree-like resolution is quasi-automatizable, meaning that a refutation of size quasi-polynomial with respect to the smallest possible can be found [6]. The following definition seems an appropriate concept of automatization for parameterized proof complexity.

Definition 6.1. *We say that a proof system P for a parameterized language L is* fpt-automatizable *if there exists an algorithm which for an instance $(x,k) \in L$ with a P-proof of size s finds a P-proof for (x,k) of size at most $f(k)(s \cdot |x|)^{O(1)}$ where f is some computable function.*

The notion of *quasi-automatizability* is meaningless in the context of parameterized (tree-like) resolution. Every $(F,k) \in \mathrm{PCon}$ with $|F| = n$ has a refutation of size $c \cdot \binom{n}{k+1}$ for some constant c. If $k \leq \log n$, then this is smaller than $n^{\log n}$ which is quasi-polynomial in n, otherwise $\binom{n}{k+1} \leq 2^{(k+1)^2}$ which is fpt with respect to k. Hence for any $(F,k) \in \mathrm{PCon}$ 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 already discussed in [2] where we want to prove non-automatizability for a system which is indeed quasi-automatizable. Another difficulty is that non-automatizability results in [1, 2] all use formulas which are interesting and meaningful on assignments of big weight. It does not seem possible to use those techniques in our framework.

Dag-like lower bounds for PHP. The combinatorial game gives lower bounds for tree-like resolution, and it could even be applied to stronger tree-like proof systems. Nevertheless, such techniques are insufficient for dag-like resolution lower bounds. The only techniques working in this context are either based on width lower bounds or on efficient interpolation methods. Both use restrictions. Parameterized resolution has efficient interpolation and we can easily prove appropriate width lower bounds for PHP. The main problem is that in both cases restrictions cannot be applied without falsifying the parameterized axioms (i. e., for less than k 1's you cannot use many 1's in the restriction).

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 J. Comput.*, 38(4):1347–1363, 2008.

[3] P. Beame, J. C. Culberson, D. G. Mitchell, and C. Moore. The resolution complexity of random graph 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 Foundations of Computer Science*, pages 274–282, 1996.

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

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

[9] 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.

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

[11] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[12] 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.

[13] J. L. Esteban, N. Galesi, and J. Messner. On the complexity of resolution with bounded conjunctions. *Theor. Comput. Sci.*, 321(2-3):347–370, 2004.

[14] J. Flum and M. Grohe. Describing parameterized complexity classes. *Inf. Comput.*, 187(2):291–319, 2003.

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

[16] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

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

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

[19] N. Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):482–537, 2007.

[20] B. Selman, H. A. Kautz, and D. A. McAllester. Ten challenges in propositional reasoning and search. In *Proc. 15th International Joint Conference on Artificial Intelligence*, pages 50–54, 1997.

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

Technical Appendix

The appendix contains all proofs that are omitted in the main part of the paper.

Theorem 2.7. *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 $L \in$ para-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 $L \in$ para-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:

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

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 $O(f(k)p(|x|))$. □

Lemma 5.4. *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 5.3 (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 5.3 (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. □

Lemma 5.5.

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

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 5.3 (part 2). Then $|P_j^0(t_{i,j})| \geq n - (2k+1)$.

We now show condition (2). 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 5.3 (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

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

□