Journal of Artificial Intelligence Research 76 (2023) 265-286 Submitted 11/2021; published 01/2023

Characterizing Tseitin-Formulas with Short Regular Resolution Refutations

Alexis de Colnet DECOLNET@AC.TUWIEN.AC.AT
Algorithms and Complexity Group, TU Wien
Vienna, Austria

Stefan Mengel MENGEL@CRIL.FR
Univ. Artois, CNRS
Centre de Recherche en Informatique de Lens (CRIL)
F-62300 Lens, France

Abstract

Tseitin-formulas are systems of parity constraints whose structure is described by a graph. These formulas have been studied extensively in proof complexity as hard instances in many proof systems. In this paper, we prove that a class of unsatisfiable Tseitin-formulas of bounded degree has regular resolution refutations of polynomial length if and only if the treewidth of all underlying graphs G for that class is in $O(\log |V(G)|)$. It follows that unsatisfiable Tseitin-formulas with polynomial length of regular resolution refutations are completely determined by the treewidth of the underlying graphs when these graphs have bounded degree. To prove this, we show that any regular resolution refutation of an unsatisfiable Tseitin-formula with graph G of bounded degree has length $2^{\Omega(tw(G))}/|V(G)|$, thus essentially matching the known $2^{O(tw(G))}\text{poly}(|V(G)|)$ upper bound. Our proof first connects the length of regular resolution refutations of unsatisfiable Tseitin-formulas to the size of representations of *satisfiable* Tseitin-formulas in decomposable negation normal form (DNNF). Then we prove that for every graph G of bounded degree, every DNNF-representation of every satisfiable Tseitin-formula with graph G must have size $2^{\Omega(tw(G))}$ which yields our lower bound for regular resolution.

1. Introduction

Resolution is one of the most studied propositional proof systems in proof complexity due to its naturality and its connections to practical SAT solving (Nordström, 2015; Buss & Nordström, 2021). A refutation of a CNF-formula in this system (a resolution refutation) relies uniquely on clausal resolution: in a refutation, clauses are iteratively derived by resolutions on clauses from the formula or previously inferred clauses, until reaching the empty clause indicating unsatisfiability. In this paper, we consider regular resolution which is the restriction of resolution to proofs in which, intuitively, variables which have been resolved away from a clause cannot be reintroduced later on by additional resolution steps. This fragment of resolution is known to generally require exponentially longer refutations than general resolution (Goerdt, 1993; Alekhnovich, Johannsen, Pitassi, & Urquhart, 2007; Urquhart, 2011; Vinyals, Elffers, Johannsen, & Nordström, 2020) but is still interesting since it corresponds to DPLL-style algorithms (Davis, Logemann, & Loveland, 1962; Davis & Putnam, 1960). Consequently, there is quite some work on regular resolution, see e.g. (Atserias, Bonacina,

de Rezende, Lauria, Nordström, & Razborov, 2018; Urquhart, 1987; Beck & Impagliazzo, 2013; Beame, Beck, & Impagliazzo, 2012) for a very small sample.

Tseitin-formulas are encodings of certain systems of linear equations whose structure is given by a graph (Tseitin, 1968). They have been studied extensively in proof complexity essentially since the creation of the field because they are hard instances in many settings, see e.g. (Urquhart, 1987; Ben-Sasson, 2002; Beame et al., 2012; Itsykson & Oparin, 2013; Itsykson, Riazanov, Sagunov, & Smirnov, 2021). It is known that different properties of the underlying graph characterize different parameters of their resolution refutations (Galesi, Talebanfard, & Torán, 2020; Alekhnovich & Razborov, 2011; Itsykson & Oparin, 2013). Extending this line of work, we here show that treewidth determines the length of regular resolution refutations of Tseitin-formulas: classes of Tseitin-formulas of bounded degree have polynomial length regular resolution refutations if and only if the treewidth of the underlying graphs is bounded logarithmically in their size. The upper bound for this result was already known from (Alekhnovich & Razborov, 2011) where it is shown that, for every graph G, unsatisfiable Tseitin-formulas with the underlying graph G have regular resolution refutations of length at most $2^{O(tw(G))}|V(G)|^c$ where c is a constant. We provide a matching lower bound:

Theorem 1. *Let $T(G, c)$ be an unsatisfiable Tseitin-formula where G is a connected graph with maximum degree at most Δ. The length of the smallest regular resolution refutation of $T(G, c)$ is at least $2^{\Omega(tw(G)/\Delta)}|V(G)|^{-1}$.*

There were already known lower bounds for the length of resolution refutations of Tseitin-formulas based on treewidth before. For *general* resolution, a $2^{\Omega(tw(G)^2)/|V(G)|}$ lower bound can be inferred with the classical width-length relation of (Ben-Sasson, 2002) and width bounds of (Galesi et al., 2020). This gives a tight $2^{\Omega(tw(G))}$ bound when the treewidth of G is linear in its number of vertices. For smaller treewidth, there are also bounds from (Galesi, Itsykson, Riazanov, & Sofronova, 2019) for the stronger proof system of depth-d Frege proofs which for resolution translate to bounds of size $2^{tw(G)^{\Omega(1)}}$, but since the top exponent is significantly less than 1, these results are incomparable to ours. Better bounds of $2^{\Omega(tw(G))/\log|V(G)|}$ for regular resolution that almost match the upper bound were shown in (Itsykson et al., 2021) for *regular* resolution refutations. Building on (Itsykson et al., 2021), we eliminate the division by $\log|V(G)|$ in the exponent and thus give a tight $2^{\Theta(tw(G))}$ dependence.

As in (Itsykson et al., 2021), our proof strategy follows two steps. First, we show that the problem of bounding the length of regular resolution refutations of an *unsatisfiable* Tseitin-formula can be reduced to lower bounding the size of certain representations of a *satisfiable* Tseitin-formula. Itsykson et al. in (Itsykson et al., 2021) used a similar reduction of lower bounds for regular resolution refutations to bounds on read-once branching programs (1-BP) for satisfiable Tseitin-formulas, using the classical connection between regular resolution and the search problem which, given an unsatisfiable CNF-formula and a truth assignment, returns a clause of the formula it falsifies (Krajícek, 1995). Itsykson et al. showed that there is a transformation of a 1-BP solving the search problem for an unsatisfiable Tseitin-formula into a 1-BP of pseudopolynomial size computing a satisfiable Tseitin-formula with the same underlying graph. This yields lower bounds for regular resolution from lower bounds for 1-BP computing satisfiable Tseitin-formulas which (Itsykson et al., 2021) also

shows. Our crucial insight here is that when more succinct representations are used to present the satisfiable formula, the transformation from the unsatisfiable instance can be changed to have only a polynomial instead of pseudopolynomial size increase. Concretely, the representations we use are so-called decomposable negation normal forms (DNNF) which are very prominent in the field of knowledge compilation (Darwiche, 2001) and generalize 1-BP. We show that every refutation of an unsatisfiable Tseitin-formula can be transformed into a DNNF-representation of a satisfiable Tseitin-formula with the same underlying graph with only polynomial overhead. The transformation builds on the relation between regular resolution refutations of unsatisfiable Tseitin-formulas and 1-BPs that, for each variable assignment, compute a contraint of the formula that is falsified. Sub-programs of such an 1-BP deal with unsatisfiable Tseitin-formulas over sub-graphs of the initial graph. During the transformation, the nodes of the 1-BP are visited in a bottom-up order and DNNF representing satisfiable variants of the corresponding Tseitin-formulas are incrementally constructed in the process.

In a second step, we then show for every satisfiable Tseitin-formula with an underlying graph G a lower bound of $2^{\Omega(tw(G))}$ on the size of DNNF computing the formula. To this end, we adapt techniques developed in (Bova, Capelli, Mengel, & Slivovsky, 2016) to a parameterized setting. (Bova et al., 2016) uses rectangle covers of a function, a common tool from communication complexity, to lower bound the size of any DNNF computing the function. Our refinement takes the form of a two-player game in which the first player tries to cover the models of a function with few rectangles while the second player hinders this construction by adversarially choosing the variable partitions respected by the rectangles from a certain set of partitions. We show that this game gives lower bounds for DNNF, and consequently the aim is to show that the adversarial player can always force $2^{\Omega(tw(G))}$ rectangles in the game when playing on a Tseitin-formula with graph G. This is done by proving that any rectangle for a carefully chosen variable partition *splits* parity constraints of the formula in a way that bounds by a function of $tw(G)$ the number of models that can be covered. We show that, depending on the treewidth of G, the adversarial player can choose a partition to limit the number of models of every rectangle constructed in the game to the point that at least $2^{\Omega(tw(G))}$ of them will be needed to cover all models of the Tseitin-formula. As a consequence, we get the desired lower bound of $2^{\Omega(tw(G))}|V(G)|^{-1}$ for regular resolution refutations of Tseitin-formulas.

2. Preliminaries

Notions on graphs. We assume the reader is familiar with the fundamentals of graph theory. For a graph G, we denote by $V(G)$ its vertices and by $E(G)$ its edges. For $v \in V(G)$, $E(v)$ denotes the edges incident to v and $N(v)$ its neighbors (v is not in $N(v)$). For a subset V' of $V(G)$ we denote by $G[V']$ the sub-graph of G induced by V'.

A binary tree whose leaves are in bijection with the edges of G is called a *branch decomposition*[1]. Each edge e of a branch decomposition T induces a partition of $E(G)$ into two parts as the edge sets that appear in the two connected components of T after deletion of e. The number of vertices of G that are incident to edges in both parts of this partition is

1. We remark that often branch decompositions are defined as unrooted trees. However, it is easy to see that our definition is equivalent, so we use it here since it is more convenient in our setting.

the order of e, denoted by $order(e, T)$. The *branchwidth* of G, denoted by $bw(G)$, is defined as $bw(G) = \min_T \max_{e \in E(T)} order(e, T)$, where $\min_T$ is over all branch decompositions of G. The *linear branchwidth* of G, denoted by $bw_\ell(G)$, is defined analogously where T is restricted to *linear branch decompositions*, that is, branch decompositions where every node is either a leaf or has a child which is a leaf.

While it is convenient to work with branchwidth or linear branchwidth in our proofs, we state our main result with the more well-known *treewidth* $tw(G)$ or *pathwidth* $pw(G)$ of a graph G. This is justified by the following connections between the four measures whose proofs can be found in (Harvey & Wood, 2017, Lemma 12), and (Nordstrand, 2017, Theorem 6.1), respectively.

Lemma 1. *If $bw(G) \geq 1$, then $bw(G) - 1 \leq tw(G) \leq \frac{3}{2} bw(G)$, and if $bw_\ell(G) \geq 1$, then $bw_\ell(G) - 1 \leq pw(G) \leq bw_\ell(G) + 1$.*

For the convenience of the reader, we reproduce the proof of the relation between linear branchwidth and pathwidth in the appendix.

A separator S in a connected graph G is defined to be a vertex set such that $G \setminus S$ is non-empty and not connected. A graph G is called 3-connected if and only if it has at least 4 vertices and, for every $S \subseteq V(G)$, $|S| \leq 2$, the graph $G \setminus S$ is connected.

Variables, assignments, v-trees. Boolean variables can have value 0 ($false$) or 1 ($true$). The notation ℓ_x refers to a literal for a variable x, that is, x or its negation $\overline{x}$. Given a set X of Boolean variables, $lit(X)$ denotes its set of literals. A truth assignment to X is a mapping $a : X \to \{0, 1\}$. If a_X and a_Y are assignments to *disjoint* sets of variables X and Y, then $a_X \cup a_Y$ denotes the combined assignment to $X \cup Y$. The set of assignments to X is denoted by $\{0, 1\}^X$. Let f be a Boolean function, we denote by $var(f)$ its variables and by $sat(f)$ its set of models, i.e., assignments to $var(f)$ on which f evaluates to 1. A v-tree of X is a binary tree T whose leaves are labeled bijectively with the variables in X. A v-tree T of X induces a set of partitions (X_1, X_2) of X as follows: choose a vertex v of T, setting X_1 to contain exactly the variables in T that appear below v and $X_2 := X \setminus X_1$.

Tseitin-formulas. Tseitin formulas are systems of parity constraints whose structure is determined by a graph. Let $G = (V, E)$ be a graph and let $c : V \to \{0, 1\}$ be a labeling of its vertices called a *charge function*. The Tseitin-formula $T(G, c)$ has for each edge $e \in E$ a Boolean variable x_e and for each vertex $v \in V$ a constraint $\chi_v : \sum_{e \in E(v)} x_e = c(v) \mod 2$. The Tseitin-formula $T(G, c)$ is then defined as $T(G, c) := \bigwedge_{v \in V} \chi_v$, i.e., the conjunction of the parity constraints for all $v \in V$. By $\overline{\chi_v}$ we denote the negation of χ_v, i.e., the parity constraint on $(x_e)_{e \in E(v)}$ with charge $1 - c(v)$.

Proposition 1. (Urquhart, 1987, Lemma 4.1) *The Tseitin-formula $T(G, c)$ is satisfiable if and only if for every connected component U of G we have $\sum_{v \in U} c(v) = 0 \mod 2$.*

Proposition 2. (Glinskih & Itsykson, 2017, Lemma 2) *Let G be a graph with K connected components. If the Tseitin-formula $T(G, c)$ is satisfiable, then it has $2^{|E(G)| - |V(G)| + K}$ models.*

When conditioning the formula $T(G, c)$ on a literal $\ell_e \in \{x_e, \overline{x_e}\}$ for $e = ab$ in $E(G)$, the resulting function is another Tseitin formula $T(G, c)|\ell_e = T(G', c')$ where G' is the graph

G without the edge e (so $G' = G - e$) and c' depends on ℓ_e. If $\ell_e = \overline{x_e}$ then c' equals c. If $\ell_e = x_e$ then $c' = c + 1_a + 1_b \mod 2$, where 1_v denotes the charge function that assigns 1 to v and 0 to all other variables.

Since we consider Tseitin-formulas in the setting of proof systems for CNF-formulas, we will assume in the following that they are encoded as CNF-formulas. In this encoding, every individual parity constraint χ_v is expressed as a CNF-formula F_v and $T(G, c) := \bigwedge_{v \in V} F_v$. Since it takes $2^{|E(v)|-1}$ clauses to write the parity constraint χ_v, each clause containing $E(v)$ literals, we make the standard assumption that $E(v)$ is bounded, i.e., there is a constant upper bound Δ on the degree of all vertices in G.

DNNF. A circuit over X in *negation normal form* (NNF) is a directed acyclic graph whose leaves are labeled with literals in $lit(X)$ or 0/1-constants, and whose internal nodes are labeled by $\vee$-gates or $\wedge$-gates. We use the usual semantics for the function computed by (gates of) Boolean circuits. Every NNF can be turned into an equivalent NNF whose nodes have at most two successors in polynomial time. So we assume that NNF in this paper have only binary gates and thus define the size $|D|$ as the number of gates, which is then linearly related to the number of wires. Given a gate g, we denote by $var(g)$ the variables for the literals appearing under g. When g is a literal input ℓ_x, we have $var(g) = \{x\}$, and when it is a 0/1-input, we define $var(g) = \emptyset$. A gate with two children g_l and g_r is called *decomposable* when $var(g_l) \cap var(g_r) = \emptyset$, and it is called *complete* (or *smooth*) when $var(g_l) = var(g_r)$. An NNF whose $\wedge$-gates are all decomposable is called a *decomposable NNF (DNNF)*. We call a DNNF *complete* when all its $\vee$-gates are complete. Every DNNF can be made complete in polynomial time. For every Boolean function f on finitely many variables, there exists a DNNF computing f.

Branching programs. A branching program (BP) B is a directed acyclic graph with a single source, sinks that uniquely correspond to the values of a finite set Y, and whose inner nodes, called *decision nodes* are each labeled by a Boolean variable $x \in X$ and have exactly two output wires called the 0- and 1-wire pointing to two nodes respectively called its 0- and 1-child. The variable x appears on a path in B if there is a decision node v labeled by x on that path. A truth assignment a to X induces a path in B which starts at the source and, when encountering a decision node for a variable x, follows the 0-wire (resp. the 1-wire) if $a(x) = 0$ (resp. $a(x) = 1$). The BP B is defined to compute the value $y \in Y$ on an assignment a if and only if the path of a leads to the sink labeled with y. We denote this value y as $B(a)$. Let $f : X \to Y$ be a function where X is a finite set of Boolean variables and Y any finite set. Then we say that B computes f if for every assignment $a \in \{0,1\}^X$ we have $B(a) = f(a)$. We say that a node v in B computes a function g if the BP we get from B by deleting all nodes that are not reachable from v computes g.

For $Y = \{0, 1\}$ we also consider *non-deterministic* BPs (NBP) which allow for additional unlabelled *guessing nodes*. In that case we define that $B(a) = 1$ if and only if there is a path from the source of B to the 1-sink following all decision nodes as for a BP and taking an arbitrary successor for guessing nodes. Note that there might a an exponential number of possible paths, and a is accepted if a single such path goes to the 1-sink. Clearly the BP are particular cases of NBP without guessing nodes.

An NBP is called *read-once*, denoted 1-NBP, when on every path each variable of X appears at most once. A 1-NBP is said to be *complete* when each variable of X appears

exactly once on every path. (Complete) 1-BP are defined as (complete) 1-NBP that are BP, i.e., that have no guessing nodes. 1-BP are also called free binary decision diagrams (FBDD) in the literature. It is well known that every 1-NBP can be converted into DNNF in polynomial time by a simple syntactical rewriting (Darwiche & Marquis, 2002; Amarilli, Capelli, Monet, & Senellart, 2020).

When representing Tseitin-formulas by DNNF or 1-NBP, we will use the following:

Lemma 2. *Let G be a graph and let c and c' be two charge functions such that $T(G, c)$ and $T(G, c')$ are satisfiable Tseitin-formulas. Then $T(G, c)$ can be computed by a DNNF (resp. 1-NBP) of size s if and only if this is true for $T(G, c')$.*

Proof. (sketch) $T(G, c)$ can be transformed into $T(G, c')$ by substituting some variables by their negations, see (Itsykson et al., 2021, Lemma 2.3). So every DNNF for $T(G, c)$ can be transformed into one for $T(G, c')$ by making the same substitutions. Replacing some variables by their negations is also feasible on 1-NBP without size increase, it boils down to inverting the 0- and 1-child of the variables in question. □

Let $R \subseteq \{0,1\}^X \times Y$ be a relation where Y is again finite. Then we say that a BP B computes R if for every assignment a we have that $(a, B(a)) \in R$. Let $T(G, c)$ be an unsatisfiable Tseitin-formula for a graph $G = (V, E)$. Then we define the two following relations: $\text{Search}_{T(G,c)}$ consists of the pairs (a, C) such that a is an assignment to $T(G, c)$ that does not satisfy the clause C of $T(G, c)$. The relation $\text{SearchVertex}(G, c)$ consists of the pairs (a, v) such that a does not satisfy the parity constraint χ_v of a vertex $v \in V$. Note that $\text{Search}_{T(G,c)}$ and $\text{SearchVertex}(G, c)$ both give a reason why an assignment a does not satisfy $T(G, c)$ but the latter is more coarse: $\text{SearchVertex}(G, c)$ only gives a constraint that is violated while $\text{Search}_{T(G,c)}$ gives an exact clause that is not satisfied.

Regular resolution. We only introduce some minimal notions of proof complexity here; for more details and references the reader is referred to the recent survey (Buss & Nordström, 2021). Let $C_1 = x \vee D_1$ and $C_2 = \overline{x} \vee D_2$ be two clauses such that D_1, D_2 contain neither x nor $\overline{x}$. Then the clause $D_1 \vee D_2$ is inferred by resolution of C_1 and C_2 on x. A resolution refutation of length s of a CNF-formula F is defined to be a sequence $C_1, \ldots, C_s$ such that C_s is the empty clause and for every $i \in [s]$ we have that C_i is a clause of F or it is inferred by resolution of two clauses C_j, C_ℓ such that $j, \ell < i$. It is well-known that F has a resolution refutation if and only if F is unsatisfiable.

To every resolution refutation $C_1, \ldots, C_s$ we assign a directed acyclic graph G as follows: the vertices of G are the clauses $\{C_i \mid i \in [s]\}$. Moreover, there is an edge C_jC_i in G if and only if C_i is inferred by resolution of C_j and some other clause C_ℓ on a variable x in the refutation. We also label the edge C_jC_i with the variable x. Note that there might be two pairs of clauses C_j, C_ℓ and $C_{j'}, C_{\ell'}$ such that resolution on both pairs leads to the same clause C_i. If this is the case, we simply choose one of them to make sure that all vertices in G have indegree at most 2. A resolution refutation is called *regular* if on every directed path in G every variable x appears at most once as a label of an edge. It is known that there is a resolution refutation of F if and only if a regular resolution refutation of F exists (Davis & Putnam, 1960), but the latter are in general longer (Alekhnovich et al., 2007; Urquhart, 2011).

In this paper, we will not directly deal with regular resolution proofs thanks to the following well-known result.

Theorem 2. (Krajícek, 1995) *For every unsatisfiable CNF-formula F, the length of the shortest regular resolution refutation of F is the size of the smallest* 1*-BP computing* Search_F.

Since in our setting, from an unsatisfied clause we can directly inferred an unsatisfied parity constraint, we can use the following simple consequence.

Corollary 1. *For every unsatisfiable Tseitin-formula $T(G, c)$, the length of the shortest regular resolution refutation of $T(G, c)$ is at least the size of the smallest* 1*-BP computing* $\text{SearchVertex}(G, c)$.

3. Reduction From Unsatisfiable to Satisfiable Formulas

To show our main result, we give a reduction from unsatisfiable to satisfiable Tseitin-formulas as in (Itsykson et al., 2021). There it was shown that, given a 1-BP B computing $\text{SearchVertex}(G, c)$ for an unsatisfiable Tseitin-formula $T(G, c)$, one can construct a 1-BP B' computing the function of a *satisfiable* Tseitin-formula $T(G, c^*)$ such that $|B'|$ is quasipolynomial in $|B|$. Then good lower bounds on the size of B' yield lower bounds for regular refutation by Corollary 1. To give tighter results, we give a version of the reduction from unsatisfiable to satisfiable Tseitin-formulas where the target representation for $T(G, c^*)$ is not 1-BP but the more succinct DNNF. This lets us decrease the size of the representation from pseudopolynomial to polynomial which, with tight lower bounds in the later parts of the paper, will yield Theorem 1.

Theorem 3. *Let $T(G, c)$ be an unsatisfiable Tseitin-formula where G is connected and let S be the length of its smallest resolution refutation. Then there exists for every satisfiable Tseitin-formula $T(G, c^*)$ a DNNF of size $O(S \times |V(G)|)$ computing it.*

In the proof of Theorem 3, we heavily rely on results from (Itsykson et al., 2021) in particular the notion of well-structuredness that we present in Section 3.1. In Section 3.2 we will then prove Theorem 3.

3.1 Well-Structured Branching Programs for SearchVertex(G, c)

In a well-structured 1-BP computing $\text{SearchVertex}(G, c)$, every decision node u_k for a variable x_e computes $\text{SearchVertex}(G_k, c_k)$ where G_k is a *connected* sub-graph of G containing the edge $e := ab$, and c_k is a charge function such that $T(G_k, c_k)$ is unsatisfiable. Since u_k deals with $T(G_k, c_k)$, its 0- and 1-successors u_{k_0} and u_{k_1} work on $T(G_k, c_k)|\ell_e$ for $\ell_e = \overline{x_e}$ and $\ell_e = x_e$, respectively. $T(G_k, c_k)|\ell_e$ is a Tseitin-formula whose underlying graph is $G_k - e$ and whose charge function is c_k or $c_k + 1_a + 1_b \mod 2$ depending on ℓ_e. For convenience, we introduce the notation $\gamma_k(x_e) = c_k + 1_a + 1_b \mod 2$ and $\gamma_k(\overline{x_e}) = c_k$. Since G_k is connected, $G_k - e$ has at most two connected components. Let G_k^a and G_k^b denote the components of $G_k - e$ containing a and b, respectively. Note that $G_k^a = G_k^b$ when e is not a bridge of G_k. Let $\gamma_k^a(\ell_e)$ and $\gamma_k^b(\ell_e)$ denote the restriction of $\gamma_k(\ell_e)$ to the vertices of G_k^a and G_k^b, respectively. While the graph for $T(G_k, c_k)|\ell_e$ has at most two connected components, exactly one of them holds an odd total charge, so only the Tseitin-formula corresponding to that

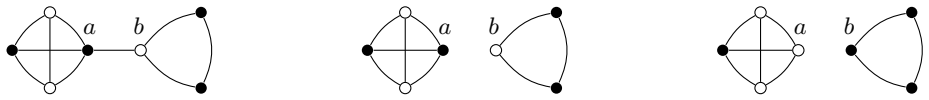

Figure 1: The graphs of Example 1. On the left the graph G_k, in the middle the result after assigning 0 to x_e, on the right after assigning 1 to x_e.

component is unsatisfiable. Well-structuredness states that u_{k_0} and u_{k_1} each deal with that unique connected component.

Example 1. Consider the graph G_k shown on the left in Figure 1. Black nodes have charge 0 and white nodes have charge 1. The corresponding Tseitin-formula $T(G_k, c_k)$ is unsatisfiable because there is an odd number of white nodes. Let $e := ab$. Then $T(G_k, c_k)|\overline{x_e}$ is the Tseitin-formula for the graph $G_k - e$ with charges as shown in the middle of Figure 1. Note that $T(G_k, c_k)|\overline{x_e}$ is unsatisfiable because of the charges in the triangle component G_k^b. The repartition of charges for $T(G_k, c_k)|x_e$ illustrated on the right of Figure 1 shows that $T(G_k, c_k)|x_e$ is unsatisfiable because of the charges in the rombus component G_k^a. Well-structuredness will ensure that, if u_k computes SearchVertex(G_k, c_k) and decides x_e, then u_{k_0} computes SearchVertex$(G_k^b, \gamma_k^b(\overline{x_e}))$ and u_{k_1} computes SearchVertex$(G_k^a, \gamma_k^a(x_e))$.

Definition 1. Let $T(G, c)$ be an unsatisfiable Tseitin-formula where G is a connected graph. A branching program B computing SearchVertex(G, c) is *well-structured* when, for all nodes u_k of B, there exists a connected subgraph G_k of G and a charge function c_k such that $T(G_k, c_k)$ is unsatisfiable, u_k computes SearchVertex(G_k, c_k), and

1. if u_k is the source, then $G_k = G$ and $c_k = c$,
2. if u_k is a sink corresponding to $v \in V(G)$, then $G_k = (\{v\}, \emptyset)$ and $c_k = 1_v$,
3. if u_k is a decision node for x_{ab} with 0- and 1- successors u_{k_0} and u_{k_1}, set $\ell_0 = \overline{x_{ab}}$ and $\ell_1 = x_{ab}$, then for all $i \in \{0, 1\}$, $(G_{k_i}, c_{k_i}) = (G_k^a, \gamma_k^a(\ell_i))$ if $T(G_k^a, \gamma_k^a(\ell_i))$ is unsatisfiable, otherwise $(G_{k_i}, c_{k_i}) = (G_k^b, \gamma_k^b(\ell_i))$.

We remark that our definition is a slight simplification of that given by Itsykson et al. (Itsykson et al., 2021). It can easily be seen that ours is implied by theirs (see Definition 3.2 and Proposition 3.4 in (Itsykson et al., 2021)).

Lemma 3. (Itsykson et al., 2021, Lemma 1.4) *Let $T(G, c)$ be an unsatisfiable Tseitin-formula where G is connected and let B be a 1-BP of minimal size computing the relation* SearchVertex(G, c). *Then B is well-structured.*

3.2 Constructing DNNF from Well-Structured Branching Programs

Similarly to Theorem 14 in (Itsykson et al., 2021), we give a reduction from a well-structured 1-BP for SearchVertex(G, c) to a DNNF computing a *satisfiable* formula $T(G, c^*)$.

Lemma 4. *Let G be a connected graph. Let $T(G, c^*)$ and $T(G, c)$ be Tseitin-formulas where $T(G, c^*)$ is satisfiable and $T(G, c)$ unsatisfiable. For every well-structured 1-BP B computing* SearchVertex(G, c) *there exists a DNNF of size $O(|B| \times |V(G)|)$ computing $T(G, c^*)$.*

Proof. Let $S = |B|$ and denote by $u_1, \ldots, u_S$ the nodes of B such that if u_j is a successor of u_i, then $j < i$ (thus u_S is the source of B). For every $i \in [S]$, the node u_i computes SearchVertex(G_i, c_i). We will show how to iteratively construct DNNF $D_1, \ldots, D_S$ such that, $D_1 \subseteq D_2 \subseteq \cdots \subseteq D_S$ and, for every $i \in [S]$,

$$\text{for all } v \in V(G_i), \text{ there is a gate } g_v \text{ in } D_i \text{ computing } T(G_i, c_i + 1_v). \qquad (*)$$

Observe that, since $T(G_i, c_i)$ is unsatisfiable, $T(G_i, c_i + 1_v)$ is satisfiable for any $v \in V(G_i)$. We show by induction on i how to construct D_i by extending D_{i-1} while respecting $(*)$.

For the base case, u_1 is a sink of B, so it computes SearchVertex$(G_v, 1_v)$ where $G_v := (\{v\}, \emptyset)$ for a vertex $v \in V(G)$. Thus we define D_1 as a single constant-1-node which indeed computes $T(G_v, 1_v + 1_v) = T(G_v, 0)$. So D_1 is a DNNF respecting $(*)$.

Now for the inductive case, suppose we have the DNNF D_{k-1} satisfying $(*)$. Consider the node u_k of B. If u_k is a sink of B, then we argue as for D_1 but since we already have the constant-1-node in D_{k-1} we define $D_k := D_{k-1}$.

Now assume that u_k is a decision node for the variable x_e with 0- and 1-successors u_{k_0} and u_{k_1}. Recall that u_k computes SearchVertex(G_k, c_k) and let $e = ab$. There are two cases. If e is not a bridge in G_k then $G_k^a = G_k^b = G_k - e$ and, by well-structuredness, u_{k_0} computes SearchVertex$(G_k - e, c_k)$ and u_{k_1} computes SearchVertex$(G_k - e, c_k + 1_a + 1_b)$. For every $v \in V(G_k)$, since $k_0, k_1 < k$, by induction there is a gate g_v^0 in D_{k_0} computing $T(G_k - e, c_k + 1_v)$ and a gate g_v^1 in D_{k_1} computing $T(G_k - e, c_k + 1_a + 1_b + 1_v)$. So for every $v \in V(G_k)$ we add to D_{k-1} an $\vee$-gate g_v whose left input is $\overline{x_e} \wedge g_v^0$ and whose right input is $x_e \wedge g_v^1$. By construction, g_v computes $T(G_k, c_k + 1_v)$ and the new $\wedge$-gates are decomposable since e is not an edge of $G_k - e$ and therefore x_e and $\overline{x_e}$ do not appear in D_{k_0} and D_{k_1}.

Now if $e = ab$ is a bridge in G_k, by well-structuredness, there exist $i \in \{0, 1\}$ and a literal $\ell_e \in \{\overline{x_e}, x_e\}$ such that u_{k_i} computes SearchVertex$(G_k^a, \gamma_k^a(\ell_e))$ and $u_{k_{1-i}}$ computes SearchVertex$(G_k^b, \gamma_k^b(\overline{\ell_e}))$. We construct a gate g_v computing $T(G_k, c_k + 1_v)$ for each $v \in V(G_k)$. Assume, without loss of generality, that $v \in V(G_k^a)$, then

- $T(G_k, c_k + 1_v)|\overline{\ell_e} \equiv T(G_k^a, \gamma_k^a(\overline{\ell_e}) + 1_v) \wedge T(G_k^b, \gamma_k^b(\overline{\ell_e})) \equiv 0$
  (because of the second conjunct which is known to be unsatisfiable), and
- $T(G_k, c_k + 1_v)|\ell_e \equiv T(G_k^a, \gamma_k^a(\ell_e) + 1_v) \wedge T(G_k^b, \gamma_k^b(\ell_e))$

For the second item, since $k_0, k_1 < k$, by induction there is a gate g_v^i in D_{k_i} computing $T(G_k^a, \gamma_k^a(\ell_e) + 1_v)$ and there is a gate g_b^{1-i} in $D_{k_{1-i}}$ computing $T(G_k^b, \gamma_k^b(\overline{\ell_e}) + 1_b)$. But $\gamma_k(\ell_e) = \gamma_k(\overline{\ell_e}) + 1_a + 1_b \mod 2$, so $\gamma_k^b(\ell_e) = \gamma_k^b(\overline{\ell_e}) + 1_b \mod 2$, therefore g_b^{i-1} computes the formula $T(G_k^b, \gamma_k^b(\ell_e))$. So we add an $\wedge$-gate g_v whose left input is ℓ_e and whose right input is $g_v^i \wedge g_b^{1-i}$ and add it to D_{k-1}. Note that $\wedge$-gates are decomposable since G_k^a and G_k^b share no edge and therefore D_{k_0} and D_{k_1} are on disjoint sets of variables.

Let D_k be the circuit after all g_v have been added to D_{k-1}. It is a DNNF satisfying both $D_{k-1} \subseteq D_k$ and $(*)$.

It only remains to bound $|D_S|$. To this end, observe that when constructing D_k from D_{k-1} we add at most $3 \times |V_k|$ gates, so $|D_S|$ is at most $3(|V_1| + \cdots + |V_S|) = O(S \times |V(G)|)$. D_S may have several gates that have no parents, call them the *roots* of D_S. Take any root of D_S and delete all gates not reached from it, the resulting circuit is a DNNF D computing a satisfiable Tseitin formula $T(G, c')$. We get a DNNF computing $T(G, c^*)$ using Lemma 2. □

Combining Corollary 1, Lemma 3 and Lemma 4 yields Theorem 3.

4. Adversarial Rectangle Bounds

In this section, we introduce the game we will use to show DNNF lower bounds for Tseitin formulas. It is based on combinatorial rectangles, a basic object of study from communication complexity.

Definition 2. A *(combinatorial) rectangle* for a variable partition (X_1, X_2) of a variables set X is defined to be a set of assignments of the form $R = A \times B$ where $A \subseteq \{0,1\}^{X_1}$ and $B \subseteq \{0,1\}^{X_2}$. The partition is called balanced when $\frac{|X|}{3} \leq |X_1|, |X_2| \leq \frac{2|X|}{3}$.

A rectangle on variables X may be seen as a function whose satisfying assignments are exactly the $a \cup b$ for $a \in A$ and $b \in B$, so we interpret rectangles as Boolean functions whenever it is convenient.

Definition 3. Let f be a Boolean function. A *balanced rectangle cover* of f is a collection $\mathcal{R} = \{R_1, \ldots, R_K\}$ of rectangles on $var(f)$ with respect to (possibly different) balanced partitions of $var(f)$, such that f is equivalent to $\bigvee_{i=1}^{K} R_i$. The minimum number of rectangles in a balanced cover of f is denoted by $R(f)$.

Theorem 4. (Bova et al., 2016, Theorem 6) *Let D be a DNNF computing a function f, then $R(f) \leq |D|$.*

The link between DNNF and balanced rectangle covers of Theorem 4 is established by analyzing particular sub-circuits of the DNNF called its *proof trees* (or *certificates* in (Bova et al., 2016)).

Definition 4. Let D be a DNNF. The *proof trees* of D defined to be tree-like sub-circuits of D that can be constructed iteratively as follows: we start from the output gate of D and add it to the proof tree. Whenever an $\wedge$-gate is met, both its child gates are added to the proof tree. Whenever a $\vee$-gate is met, exactly one child is added to the proof tree.

Each proof tree of a DNNF computes a conjunction of literals. By distributivity, the following holds.

Proposition 3. *The disjunction of the conjunctions computed by the proof trees of a DNNF D computes the same function as D.*

Proposition 4. *In a complete DNNF, every variable appears exactly once in every proof tree. So every proof tree of a complete DNNF encodes a single model.*

When trying to show parameterized lower bounds with Theorem 4, one often runs into the problem that it is somewhat inflexible: the partitions of the rectangles in covers have to be balanced, but in parameterized applications this is often undesirable. Instead, to show good lower bounds, one wants to be able to partition in places that allow to cut in complicated subparts of the problem. This is e.g. the underlying technique in (Razgon, 2016). To make this part of the lower bound proofs more explicit and the technique more reusable, we here introduce a refinement of Theorem 4.

We define the adversarial multi-partition rectangle cover game for a function f on variables X and a set $S \subseteq sat(f)$ to be played as follows: two players, the cover player Charlotte and her adversary Adam, construct in several rounds a set $\mathcal{R}$ of combinatorial rectangles that cover the set S respecting f (that is, rectangles in $\mathcal{R}$ contain only models of f). The game starts with $\mathcal{R}$ as the empty set. Charlotte starts a round by choosing a v-tree T of X. Now Adam chooses a partition (X_1, X_2) of X induced by T. Charlotte ends the round by adding to $\mathcal{R}$ a combinatorial rectangle for this partition and respecting f. The game is over when S is covered by $\mathcal{R}$. The adversarial multi-partition rectangle complexity of f and S, denoted by $aR(f, S)$ is the minimum number of rounds in which Charlotte can finish the game, whatever the choices of Adam are. The following theorem gives the core technique for showing lower bounds later on. The linear adversarial multi-partition rectangle complexity $aR_\ell(f, S)$ of f and S is defined analogously with the difference that instead of a v-tree Charlotte gives an order of X and Adam chooses (X_1, X_2) such that X_1 is a prefix of the order given by Charlotte.

Theorem 5. *Let D be a complete DNNF computing a function f and let $S \subseteq sat(f)$. Then $aR(f, S) \leq |D|$.*

Proof. Let $X = var(D)$. We iteratively delete vertices from D and construct rectangles. Let $\mathcal{R}$ be the rectangle cover, initially empty, during the game. The approach is as follows: Charlotte chooses a proof tree T in D that accepts some assignment in S not yet covered by $\mathcal{R}$. By completeness of D, all variables of X appear exactly once in T. Charlotte constructs a v-tree of X from T by deleting negations on the leaves, contracting away nodes with a single child and forgetting the labels of all operation gates. Now Adam chooses a partition induced by T given by a subtree of T with root v. Note that v is a gate of C. Let $sat(D, v) \subseteq sat(f)$ be the assignments to X accepted by a proof tree of C passing through v, and observe that $sat(D, v)$ is a combinatorial rectangle $A \times B$ with $A \subseteq \{0,1\}^{var(v)}$ and $B \subseteq \{0,1\}^{X \setminus var(v)}$. Charlotte chooses the rectangle $sat(D, v)$, add it to $\mathcal{R}$ and the game continues.

Note that the vertex v in the above construction is different for every iteration of the game: by construction, Charlotte never chooses a proof tree that passes through a vertex v that has appeared before since $sat(D, v)$ is covered by $\mathcal{R}$. Consequently, the game will never last more than $|D|$ rounds. □

If the input is an 1-BP instead of a DNNF, the proof trees chosen by Charlotte are accepting paths in D, inducing an order of X. Hence, we get the following corollary.

Corollary 2. *Let B be a complete 1-NBP computing a function f and let $S \subseteq sat(f)$. Then $aR_\ell(f, S) \leq |B|$.*

When $S = sat(f)$, we will just write $aR(f)$ for $aR(f, sat(f))$ and $aR_\ell(f)$ for $aR_\ell(f, sat(f))$.

5. Splitting Parity Constraints

In this section, we will see that rectangles *split* parity constraints in a certain sense and show how this is reflected in in the underlying graph of Tseitin-formulas. This will be crucial in proving the DNNF lower bound in the next section with the adversarial multi-partition rectangle cover game.

5.1 Rectangles Induce Sub-Constraints for Tseitin-Formulas

Let R be a rectangle for the partition (E_1, E_2) of $E(G)$ such that $R \subseteq sat(T(G,c))$. Assume that there is a vertex v of G incident to edges in E_1 and to edges in E_2, i.e., $E(v) = E_1(v) \cup E_2(v)$ where neither $E_1(v)$ not $E_2(v)$ is empty. We will show that R does not only respect χ_v, but it also respects a sub-constraint of χ_v.

Definition 5. Let χ_v be a parity constraint on $(x_e)_{e\in E(v)}$. A sub-constraint of χ_v is a parity constraint χ'_v on a non-empty proper subset of the variables of χ_v.

Lemma 5. *Let $T(G,c)$ be a satisfiable Tseitin-formula and let R be a rectangle for the partition (E_1, E_2) of $E(G)$ with $R \subseteq sat(T(G,c))$. If $v \in V(G)$ is incident to edges in E_1 and to edges in E_2, then there exists a sub-constraint χ'_v of χ_v such that $R \subseteq sat(T(G,c) \wedge \chi'_v)$.*

Proof. Let $a_1 \cup a_2 \in R$ where a_1 is an assignment to E_1 and a_2 an assignment to E_2. Let $a_1(v)$ and $a_2(v)$ denote the restriction of a_1 and a_2 to $E_1(v)$ and $E_2(v)$, respectively. We claim that for all $a'_1 \cup a'_2 \in R$, we have that $a'_1(v)$ and $a_1(v)$ have the same parity, that is, $a_1(v)$ assigns an odd number of variables of $E_1(v)$ to 1 if and only if it is also the case for $a'_1(v)$. Indeed if $a_1(v)$ and $a'_1(v)$ have different parities, then so do $a_1(v) \cup a_2(v)$ and $a'_1(v) \cup a_2(v)$. So either $a_1 \cup a_2$ or $a'_1 \cup a_2$ falsifies χ_v, but both assignments are in R, so $a_1(v)$ and $a'_1(v)$ cannot have different parities as this contradicts $R \subseteq sat(T(G,c))$. Let c_1 be the parity of $a_1(v)$, then we have that assignments in R must satisfy $\chi'_v : \sum_{e\in E_1(v)} x_e = c_1 \mod 2$, so $R \subseteq sat(T(G,c) \wedge \chi'_v)$. □

Renaming χ'_v as χ^1_v and adopting notations from the proof, one sees that $\chi^1_v \wedge \chi_v \equiv \chi^1_v \wedge \chi^2_v$ where $\chi^2_v : \sum_{e\in E_2(v)} x_e = c(v) + c_1 \mod 2$. So R respects the formula $(T(G,c) - \chi_v) \wedge \chi^1_v \wedge \chi^2_v$ where $(T(G,c) - \chi_v)$ is the formula obtained by removing all clauses of χ_v from $T(G,c)$. In this sense, the rectangle is splitting the constraint χ_v into two subconstraints in disjoint variables. Since $\chi_v \equiv (\chi^1_v \wedge \chi^2_v) \vee (\overline{\chi}^1_v \wedge \overline{\chi}^2_v)$ it is plausible that potentially many models of χ_v are not in R. We show that this is true in the next section.

5.2 Vertex Splitting and Sub-Constraints for Tseitin-Formulas

Let $v \in V(G)$ and let (N_1, N_2) be a proper partition of $N(v)$, that is, neither N_1 nor N_2 is empty. The graph G' we get by *splitting* v along (N_1, N_2) is defined as the graph we get by deleting v, adding two vertices v^1 and v^2, and connecting v^1 to all vertices in N_1 and v^2 to all vertices in N_2. We now show that splitting a vertex v in a graph G has the same effect as adding a sub-constraint of χ_v.

Lemma 6. *Let $T(G,c)$ be a Tseitin-formula. Let $v \in V(G)$ and let (N_1, N_2) be a proper partition of $N(v)$. Let c_1 and c_2 be such that $c_1 + c_2 = c(v) \mod 2$ and let $\chi^i_v : \sum_{u\in N_i} x_{uv} = c_i \mod 2$ for $i \in \{1,2\}$ be sub-constraints of χ_v. Call G' the result of splitting v along (N_1, N_2) and set*

$$c'(u) := \begin{cases} c(u), & \text{if } u \in V(G) \setminus \{v\} \\ c_i, & \text{if } u = v^i, i \in \{1,2\} \end{cases}$$

There is a bijection $\rho : var(T(G,c)) \to var(T(G',c'))$ acting as a renaming of the variables such that $T(G',c') \equiv (T(G,c) \wedge \chi^1_v) \circ \rho$.

Proof. Denote by $T(G,c) - \chi_v$ the formula equivalent to the conjunction of all χ_u for $u \in V(G) \setminus \{v\}$. Then $T(G,c) \wedge \chi_v^1 \equiv (T(G,c) - \chi_v) \wedge \chi_v^1 \wedge \chi_v^2$. The constraints χ_u for $u \in V(G) \setminus \{v\}$ appear in both $T(G',c')$ and in $T(G,c) - \chi_v$ and the sub-constraints χ_v^1 and χ_v^2 are exactly the constraints for v^1 and v^2 in $T(G',c')$ modulo the variable renaming ρ defined by $\rho(x_{uv}) = x_{uv^1}$ when $u \in N_1$, $\rho(x_{uv}) = x_{uv^2}$ when $u \in N_2$, and $\rho(x_e) = x_e$ when v is not incident to e.

□

Intuitely, Lemma 6 says that splitting a vertex in G and adding sub-constraint are essentially the same operation. This allows us to compute the number of models of a Tseitin-formula to which a sub-constraint was added.

Lemma 7. *Let $T(G,c)$ be a satisfiable Tseitin-formula where G is connected. Define $T(G',c')$ as in Lemma 6. If G' is connected then $T(G',c')$ has $2^{|E(G)|-|V(G)|}$ models.*

Proof. $T(G,c)$ is satisfiable and $\sum_{u \in V(G')} c'(u) = \sum_{u \in V(G)} c(u) = 0 \mod 2$ so $T(G',c')$ is satisfiable by Proposition 1. Using Proposition 2 yields that $T(G',c')$ has $2^{|E(G')|-|V(G')|+1} = 2^{|E(G)|-|V(G)|}$ models. □

Lemma 8. *Let $T(G,c)$ be a satisfiable Tseitin-formula where G is connected. Let $\{v_1,\dots,v_k\}$ be an independent set in G. For all $i \in [k]$ let (N_1^i, N_2^i) be a proper partition of $N(v_i)$ and let $\chi'_{v_i} : \sum_{u \in N_1^i} x_{uv_i} = c_i \mod 2$. If the graph obtained by splitting all v_i along (N_1^i, N_2^i) is connected, then the formula $T(G,c) \wedge \chi'_{v_1} \wedge \dots \wedge \chi'_{v_k}$ has $2^{|E(G)|-|V(G)|-k+1}$ models.*

Proof. An easy induction based on Lemma 6 and Lemma 7. The induction works since, $\{v_1,\dots,v_k\}$ being an independant set, the edges to modify by splitting v_i are still in the graph where $v_1,\dots,v_{i-1}$ have been split. □

5.3 Vertex Splitting in 3-Connected Graphs

When we want to apply the results of the last sections to bound the size of rectangles, we require that the graph G remains connected after splitting vertices. This is obviously not true for all choices of vertex splits, but here we will see that if G is sufficiently connected, then we can always chose a large subset of any set of potential splits such that, after applying the split for this subset, G remains connected.

Lemma 9. *Let G be a 3-connected graph of and let $I = \{v_1,\dots,v_k\}$ be an independent set in G. For every $i \in [k]$ let (N_1^i, N_2^i) be a proper partition of $N(v_i)$. Then there is a subset $S \subseteq I$ of size at least $k/3$ such that the graph resulting from splitting all $v_i \in S$ along the corresponding (N_1^i, N_2^i) is connected.*

Proof. Let $C_1,\dots,C_r$ be the connected components of the graph G_1 that we get by splitting *all* v_i. If G_1 is connected, then we can set $S = I$ and we are done. So assume that $r > 1$ in the following. Now add for every $i \in [k]$ the edge (v_i^1, v_i^2). Call this edge set L (for *links*) and the resulting graph G_2. Note that G_2 is connected and for every edge set $E' \subseteq L$ we have that $G_2 \setminus E'$ is connected if and only if G is connected after splitting the vertices corresponding to the edges in E'. Denote by L_{in} the edges in L whose end points both lie in some component C_j and let $L_{out} := L \setminus L_{in}$. Note that $2k = 2|I| = |L_{out}| + |L_{in}|$.

Let $S^j = \{v \in I \mid |\{v^1, v^2\} \cap V(C_j)| = 1\}$ be the subset of I such that after splitting the vertices in I, exactly one side of the vertex ends up in C_j. We show that $|S^j| \geq 3$. Since G_2 is connected but the set C_j is a connected component of $G_2 \setminus L = G_1$, there must be at least one edge in L incident to a vertex in C_j. By construction this vertex is in I, say it is v_i. Since $N_1^i \neq \emptyset$ and $N_2^i \neq \emptyset$, we have that v_i has a neighbor w in C_j and $w \notin I$ (since I is an independent set). If we delete the vertices of S^j, then a subset of C_j becomes disconnected from the rest of G^2 (which is non-empty because there is at least one component different from C_j in G_2 which also contains a vertex not in I by the same reasoning as before). But then, because G is 3-connected, there must be at least three vertices in S^j.

Now we have that $2|L_{out}| = \sum_{i=1}^{r} |S^j| \geq 3r$, so

$$r \leq \frac{2}{3}|L_{out}|.$$

Now contract all components C_j in G_2 and call the resulting multigraph G_3. Note that G_3 is connected, that $|V(G_3)| = r$, and that $E(G_3) = L_{out}$. Let E_T be the edges of a spanning tree of G_3 and let E^* be the remaining edges. Then $|L_{out}| = |E_T| + |E^*| = r - 1 + |E^*| < 2|L_{out}|/3 + |E^*|$ and thus

$$|E^*| > \frac{|L_{out}|}{3}.$$

Recall that each edge in $L_{out} = E(G_3)$ corresponds to a vertex in I that was split when going from G to G_1. Since G_3 remains connected when we delete the edge set E^*, splitting only the vertices corresponding to E^* in G yields a connected graph. Finally, the number of vertices of I that we can split in G while preserving connectivity is the same as the number of links in G_2 that can be deleted while preserving connectivity. If we choose not to delete the edges that, after contradiction of G_2 to G_3 constitute E_T, then the number of links of G_2 that we can delete while preserving connectivity is,

$$|L_{in} \cup E^*| = |L_{in}| + |E^*| = |L| - |L_{out}| + |E^*| > |L| - \frac{2}{3}|L_{out}| \geq \frac{1}{3}|L| = \frac{k}{3}.$$

□

6. DNNF Lower Bounds for Tseitin-Formulas

In this section, we use the results of the previous sections to show our lower bounds for DNNF computing Tseitin-formulas. To this end, we first show that we can restrict ourselves to the case of 3-connected graphs.

6.1 Reduction from Connected to 3-Connected Graphs

In (Bodlaender & Koster, 2006), Bodlaender and Koster study how separators can be used in the context of treewidth. They call a separator S *safe for treewidth* if there exists a connected component of $G \setminus S$ whose vertex set V' is such that $tw(G[S \cup V'] + clique(S)) = tw(G)$, where $G[S \cup V'] + clique(S)$ is the graph induced on $S \cup V'$ with additional edges that pairwise connect all vertices in S.

Lemma 10. (Bodlaender & Koster, 2006, Corollary 15) *Every separator of size 1 is safe for treewidth. When G has no separator of size 1, every separator of size 2 is safe for treewidth.*

Remember that a *topological minor* H of a G is a graph that can be constructed from G by iteratively applying the following operations:

- edge deletion,
- deletion of isolated vertices, or
- subdivision elimination: if $\deg(v) = 2$ then delete v and, if its two neighbors are not already connected, then connect them.

Note that for subdivision elimination, if the two neighbors of v are already connected, then the subdivision elimination boils down to two edge deletions followed by the deletion of the now isolated vertex v.

Lemma 11. *Let H be a topological minor of G. If the satisfiable Tseitin-formula $T(G, 0)$ has a DNNF of size s, then so does $T(H, 0)$.*

Proof. Let D be a DNNF representing $T(G, 0)$. We show how to obtain a DNNF D' representing $T(H, 0)$ with $|D'| = |D|$ when H is obtained by applying a single operation (edge deletion, isolated vertex deletion, subdivision elimination). The lemma will then follow by induction.

If H is obtained by deleting an isolated vertex v from G, then $T(H, 0) = T(G, 0)$ since isolated vertices give no constraints and thus no clauses in $T(G, 0)$. So in this case $D' = D$.

If H is obtained by deleting an edge $e = uv$ from G, then $T(H, 0) = T(G, 0)|\overline{x_e}$. Conditioning a DNNF on a variable assignment does not increase its size: one can just replace in D every occurence of x_e by 0 and every occurence $\overline{x_e}$ by 1 to obtain D'. Clearly $|D'| = |D|$.

Finally assume that there is a vertex v of degree 2 in G incident to the edges $e_1 = uv$ and $e_2 = vw$ and say that H is constructed from G by subdivision elimination of v. There are two cases. If $uw \in E(G)$, then H is obtained by removing v, e_1 and e_2 from G. In other words, H is derived from G by two edge deletions (e_1 and e_2) followed by one isolated vertex deletion (v). We have already treated these operations so we know how to obtain D' from D in this case. If however $uw \notin E(G)$ then H is obtained by deleting v, e_1, e_2 and by connecting u to w. Since $c(v) = 0$, the constraint $\chi_v : x_{e_1} + x_{e_2} = 0 \mod 2$ implies that, in every satisfying assignment, x_{e_1} and x_{e_2} take the same value. Thus, abusing notation a little and calling e_1 the edge between u and w in H, we obtain that $T(H, 0) \equiv T(G, 0)|x_{e_2} \vee T(G, 0)|\overline{x_{e_2}}$. We say that $T(H, 0)$ is obtained by *forgetting* x_{e_2} *from* $T(G, 0)$, denoted $T(H, 0) \equiv \exists x_{e_2}.T(G, 0)$. Forgetting is feasible on DNNF without size increase. Indeed forgetting x_{e_2} from D boils down to replacing *both* occurrences of x_{e_2} and occurences of $\overline{x_{e_2}}$ by 1 (Darwiche, 2001). Calling the resulting DNNF D', we clearly have $|D'| = |D|$. □

We remark in passing that a result analogous to Lemma 11 is also true for 1-NBP and 1-BP instead of DNNF. For 1-NBP this is directly clear since the forgetting operation that we use in the proof is also possible on 1-NBP without any size increase and so the same proof

works. However, for 1-BP the proof is not immediate, since for them forgetting variables generally leads to an unavoidable blow-up in size (Darwiche & Marquis, 2002). The proof can still be made to work nevertheless because we are in a special case in which the variable to forget is equivalent to a variable that remains and thus the operation is possible without any size increase. Since we do not use Lemma 11 for 1-NBP or 1-BP, we leave out the details.

Lemma 12. *Let G be a graph with treewidth at least 3. Then G has a 3-connected topological minor H with $tw(H) = tw(G)$.*

Proof. First we construct a topological minor of G with no separator of size 1 that preserves treewidth. Let $S = \{v\}$ be a separator of size 1 of G, then $G \setminus S$ has a connected component V' such that $G[S \cup V'] + clique(S) = G[S \cup V']$ has treewidth $tw(G)$. Let $G' = G[S \cup V']$, then $tw(G') = tw(G)$. Observe that G' is a topological minor (remove all edges not in $G[S \cup V']$ thus isolating all vertices not in $S \cup V'$, which are then deleted) where S is no longer a separator. Repeat the construction until G' has no separator of size 1.

Now assume $S = \{u, v\}$ is a separator of G'. If V' are the vertices of a connected component of $G' \setminus S$, then there is a path from u to v in $G[S \cup V']$ since otherwise either $\{u\}$ or $\{v\}$ is a separator of size 1 of G'. Lemma 10 ensures that there is a connected component H' in $G' \setminus S$ such that $H := (V(H') \cup S, E(G[V(H') \cup S]) \cup \{uv\})$ has treewidth $tw(H) = tw(G') = tw(G)$. Let us prove that H is topological minor of G'. Consider a connected component of $G' \setminus S$ distinct from H' with vertices V' and let P be a path connecting u to v in $G[S \cup V']$. Delete all edges of $G[S \cup V']$ not in P, then delete all isolated vertices in V' so that only P remains, finally use subdivision elimination to reduce P to a single edge uv. Repeat the procedure for all connected components of $G' \setminus S$ distinct from H', the resulting topological minor is $G[V(H') \cup S]$ with the (additional) edge uv, so H.

Repeat the construction until there are no separators of size 1 or size 2 left. Note that this process eventually terminates since the number of vertices decreases after every separator elimination. The resulting graph H is a topological minor of G of treewidth $tw(G)$ without separators of size 1 or 2. Since $tw(H) = tw(G) \geq 3$, we have that H has at least 4 vertices, so H is 3-connected. □

Lemma 12 does not hold when replacing treewidth by pathwidth. To see this, note that trees can have arbitrarily high pathwidth, see (Scheffler, 1989), and trees with more than 3 vertices are clearly not 3-connected. Connected topological minors of trees are again trees, so a topological minor of a tree is 3-connected if and only if it has at most three vertices and therefore its pathwidth is at most 2.

6.2 Proof of the DNNF Lower Bound and of the Main Result

Lemma 13. *Let $T(G, c)$ be a satisfiable Tseitin-formula where G is a connected graph with maximum degree at most Δ. Any complete DNNF computing $T(G, c)$ has size at least $2^{\Omega(tw(G)/\Delta)}$.*

Proof. By Lemma 2 we can set $c = 0$. By Lemmas 11 and 12 we can assume that G is 3-connected. The lemma is proved by invoking Theorem 5 and showing that $aR(T(G, c)) = 2^{\Omega(tw(G)/\Delta)}$. More specifically, we show that the adversarial multi-partition rectangle complexity is lower-bounded by 2^k for $k := \frac{2tw(G)}{9(\Delta+1)}$. To this end, we will show that the rectangles

that Charlotte can construct after Adam's answer are never bigger than $2^{|E(G)|-|V(G)|-k+1}$. Since $T(G,c)$ has $2^{|E(G)|-|V(G)|+1}$ models, the claim then follows.

So let Charlotte choose a v-tree T. Note that since the variables of $T(G,0)$ are the edges of G, the v-tree T is also a branch decomposition of G. Now by the definition of branchwidth, Adam can choose a cut of T inducing a partition (E_1, E_2) of $E(G)$ for which there exists a set $V' \in V(G)$ of at least $bw(G) \geq \frac{2}{3}tw(G)$ vertices incident to edges in E_1 and to edges in E_2.

G has maximum degree Δ so there is an independent set $V'' \subseteq V'$ of size at least $\frac{|V'|}{\Delta+1}$. Since G is 3-connected, by Lemma 9 there is a subset $V^* \subseteq V''$ of size at least $\frac{|V''|}{3} \geq \frac{2tw(G)}{9(\Delta+1)} = k$ such that G remains connected after splitting of the nodes in V^* along the partition of their neighbors induced by the edges partition (E_1, E_2). Using Lemma 5, we find that any rectangle R for the partition (E_1, E_2) respects a sub-constraint χ'_v for each $v \in V^*$. So R respects $T(G,0) \wedge \bigwedge_{v \in V^*} \chi'_v$. Finally, Lemma 8 shows that $|R| \leq 2^{|E(G)|-|V(G)|-k+1}$, as required. □

Theorem 1 is now a direct consequence of Theorem 3, Lemma 13 and Lemma 2. Let us however sum up the chain of inequalities that led to Theorem 1. First we have showned the following:

Regular Refutation Length for an unsatisfiable $T(G,c)$

$\geq$ 1-BP size for $\text{Search}_{T(G,c)}$	(Theorem 2)
$\geq$ 1-BP size for SearchVertex$(T(G,c))$	(Corollary 1)
$=$ well-structured 1-BP size for SearchVertex$(T(G,c))$	(Lemma 3)
$\geq$ DNNF size for $T(G,0)/O(\lvert V(G)\rvert)$	(Theorem 3, Lemma 2)

Then we have shown a bound on the size of the smallest DNNF representing a satisfiable Tseitin formula, that can be summarized as follows. Let H be the 3-connected topological minor of G chosen by Lemma 12.

DNNF size for $T(G,0)$

$\geq$ DNNF size for $T(H,0)$	(Lemma 12)
$\geq$ $aR(T(H,0))$	(Theorem 5)
$\geq$ 2^k where $k = 2\,tw(H)/(9(\Delta(H)+1))$	(Lemma 13)
$=$ 2^k where $k = 2\,tw(G)/(9(\Delta(H)+1))$	(Lemma 12)
$\geq$ 2^k where $k = 2\,tw(G)/(9(\Delta(G)+1))$	($\Delta(G) \geq \Delta(H)$)

Regarding Lemma 13, note that a similar lower bounds using pathwidth instead of treewidth holds on the size of the smallest 1-NBP computing a Tseitin-formula. But then, as explained before, the reduction to Tseitin-formulas structured by 3-connected graphs does not preserve the pathwidth. So we can only phrase our result with pathwidth for Tseitin-formulas whose graphs are already 3-connected.

Lemma 14. *Let $T(G,c)$ be a satisfiable Tseitin-formula where G is a 3-connected graph with maximum degree at most Δ. Then every 1-NBP computing $T(G,c)$ has size at least $2^{\Omega(pw(G)/\Delta)}$.*

Proof. The proof is the similar to that of Lemma 13. Again by Lemma 2 we can set $c = 0$ and this time it is assumed in the lemma's statement that G is 3-connected. The proof then follows Lemma 13's proof to show that the *linear* adversarial multi-partition rectangle complexity is lower-bounded by 2^k for $k := \frac{pw(G)}{3\Delta}$. One can just follows the second and third paragraphs of that proof, replacing vtree by *linear* vtree, branch decomposition by *linear* branch decomposition, branchwidth with *linear* branchwidth, and using the inequality $bw_\ell(G) \geq pw(G)$ from Lemma 1. □

In comparison, (Itsykson et al., 2021) introduced a new width measure for graphs called the *component width*, denoted by $compw(G)$, and proved that the 1-NBP size for *any* satisfiable Tseitin-formula $T(G, c)$ is between $2^{compw(G)}$ and $|E(G)| \times 2^{compw(G)} + 2$. They compared the component width of a graph to its treewidth and its pathwidth and were able to show that $\frac{1}{2}(tw(G) - 1) \leq compw(G) \leq pw(G) + 1$ and that the two bounds are thight. On the one hand, since the component width of trees is 0 while the pathwidth is unbounded, our Lemma 14 does not extend to cases where G is not 3-connected. On the other hand Lemma 14 also shows the component width and the pathwidth are linearly related for 3-connected graphs whose maximum degree is bounded by a constant.

7. Conclusion

We have shown that the unsatisfiable Tseitin-formulas with polynomial length of regular resolution refutations are completely determined by the treewidth of their graphs. We did this by connecting lower bounds on these types of refutations to size bounds on DNNF representations of Tseitin-formulas. Moreover, we introduced a new two-player game that allowed us to show DNNF lower bounds.

Let us discuss some questions that we think are worth exploring in the future. First, it would be interesting to see if a $2^{\Omega(tw(G))}$ lower bound for the refutation of Tseitin-formulas can also be shown for general resolution. In that case the length of resolution refutations would essentially be the same as that regular resolution refutations for Tseitin formulas. Note that this is somewhat plausible since other measures like space and width are known to be the same for the two proof systems for these formulas (Galesi et al., 2020).

Another question is the relation between knowledge compilation and proof complexity. As far as we are aware, our Theorem 3 is the first result that connects bounds on DNNF to such in proof complexity. It would be interesting to see if this connection can be strenghtened to other classes of instances, other proof systems, representations from knowledge compilation and measures on proofs and representations, respectively.

Acknowledgments

We thank the reviewers for their helpul suggestions to improve the readibility of the paper. This work has been partly supported by the PING/ACK project of the French National Agency for Research (ANR-18-CE40-0011).

Appendix A. Pathwidth and Linear Branchwidth

We prove the relation between pathwidth and linear branchwidth stated in Lemma 1. We shall use the definition of pathwidth in terms of path decomposition. Given a graph G, a path decomposition of G is a finite sequence $\mathcal{P} = (B_1, \ldots, B_N)$ of vertex sets called *bags*, such that

- $V(G) = B_1 \cup \cdots \cup B_N$
- for every $uv \in E(G)$, there is a bag B_i containing both u and v
- for every i, j and k such that $1 \leq i < j < k \leq N$, we have $B_i \cap B_k \subseteq B_j$.

The *width* of the path decomposition $\mathcal{P}$ is defined as

$$width(\mathcal{P}) := \max_{1 \leq i \leq |\mathcal{P}|} |B_i| - 1.$$

The pathwidth of G is the minimum width of a path decomposition of G, so $pw(G) := \min_{\mathcal{P}} width(\mathcal{P})$.

Recall that a branch decomposition T of G is a binary tree whose leaves are in bijection with $E(G)$. Removing any $e \in E(T)$ splits the tree into two connected components, so it also partitions the edges of $E(G)$ into two sets $E_1(e,T)$ and $E_2(e,T)$ that are in bijection with the leaves of the first and second component, respectively. For $e \in E(T)$, $order(e,T)$ is the number of vertices in $V(G)$ that are incident to some edge in $E_1(e,T)$ and to some edge in $E_2(e,T)$. Recall that the linear branchwidth is defined as $bw_\ell(G) = \min_T \max_{e \in E(T)} order(e,T)$ where T are *linear* branch decompositions. Linear branch decompositions of G can alternatively be seen as orders of the edges of G. Let $E(G) = \{e_1, \ldots, e_m\}$, let π be a permutation of $\{1, \ldots, m\}$ and consider the ordering $e_{\pi(1)}, \ldots, e_{\pi(m)}$. For all $1 \leq i < m$ we define $order(i, \pi)$ to be the number of vertices in $V(G)$ that are incident to some edge in $\{e_{\pi(1)}, \ldots, e_{\pi(i)}\}$ and to some edge in $\{e_{\pi(i+1)}, \ldots, e_{\pi(m)}\}$. It is easy to see that $bw_\ell(G) = \min_\pi \max_i order(i, \pi)$.

The following lemma is shown in the master thesis (Nordstrand, 2017).

Lemma 15. *If G has at least two edges then $pw(G) - 1 \leq bw_\ell(G) \leq pw(G) + 1$.*

Proof. Removing isolated vertices in the graph does not impact the pathwidth or the linear branchwidth, so without loss of generality we assume that G has no such vertices.

First we prove that $pw(G) - 1 \leq bw_\ell(G)$. Assume G has linear branchwidth k and let $E(G) = \{e_1, \ldots, e_m\}$. Let π be a permutation of $\{1, \ldots, m\}$. For i ranging from 1 to m define B_i to be the vertex set containing both vertices of $e_{\pi(i)}$ plus all vertices that are incident to some edge in $\{e_{\pi(1)}, \ldots, e_{\pi(i)}\}$ and to some edge in $\{e_{\pi(i+1)}, \ldots, e_{\pi(m)}\}$ (which is not empty since $m \geq 2$). Observe that if there is $i' < i < i''$ such that u is incident to $e_{\pi(i')}$ and to $e_{\pi(i'')}$ then u is in B_i. From the definition of B_i we deduce that $|B_i| \leq order(i, \pi) + 2$ (the $+2$ comes from the vertices of $e_{\pi(i)}$) so if $B_1, \ldots, B_m$ is a path decomposition then its width is at most $\max_i order(i, \pi) + 1$. We claim that $B_1, \ldots, B_m$ is a path decomposition. Fr the first two conditions and since G has no isolated vertices, we obtain that the union of $B_1, \ldots, B_m$ equals $V(G)$ and that every edge e_j of G is at least contained in $B_{\pi^{-1}(j)}$. Now

for the third condition, take $1 \leq i < j < k \leq m$ and consider $u \in B_i \cap B_k$. By definition there are $i' \leq i \leq i''$ and $k' \leq k \leq k''$ such that u is a vertex of $e_{\pi(i')}$, $e_{\pi(i'')}$, $e_{\pi(k')}$ and $e_{\pi(k'')}$. Since but then $i' < j < k''$ holds and therefore u is in B_j.

Now we prove the other direction, so $bw_\ell(G) \leq pw(G) + 1$. Consider a path decomposition $\mathcal{P} = (B_1, \ldots, B_N)$. We construct π as follows. Let $c = 1$ and start with all edges of G unmarked. We visit the bags in order. When the vertices of an unmarked edge e_i are both in the current bag, then set $\pi(c)$ to i, mark e_i and increment c. If there are several such edges, we apply an arbitrary tie-breaking and treat all of them in any order. At the end of the procedure all the edges are marked (since each edge is contained in some bag), so π is permutation of $\{1, \ldots, m\}$. Now for each $1 \leq i \leq m$, let i^* be the smallest index of a bag containing e_i. By construction if $\pi^{-1}(i) \leq \pi^{-1}(j)$ then $i^* \leq j^*$. Now take $1 \leq k < m$, and let u be a vertex incident to an edge e_i in $\{e_{\pi(1)}, \ldots, e_{\pi(\pi^{-1}(k))}\}$ and to an edge e_j in $\{e_{\pi(\pi^{-1}(k)+1)}, \ldots, e_{\pi(m)}\}$. Since $\pi^{-1}(i) \leq \pi^{-1}(k) < \pi^{-1}(j)$ we have $i^* \leq k^* \leq j^*$. Moreover, since $e_i \subseteq B_{i^*}$ and $e_j \subseteq B_{j^*}$ we have $u \in B_{i^*} \cap B_{j^*} \subseteq B_{k^*}$. This shows that $order(\pi^{-1}(k), \pi) \leq |B_{k^*}|$. So if $\mathcal{P}$ is a path decomposition of width $pw(G)$ the corresponding ordering π verifies $\max_k order(k, \pi) \leq pw(G) + 1$. □

References

Alekhnovich, M., Johannsen, J., Pitassi, T., & Urquhart, A. (2007). An Exponential Separation between Regular and General Resolution. *Theory Comput.*, *3*(1), 81–102.

Alekhnovich, M., & Razborov, A. A. (2011). Satisfiability, Branch-Width and Tseitin tautologies. *Comput. Complex.*, *20*(4), 649–678.

Amarilli, A., Capelli, F., Monet, M., & Senellart, P. (2020). Connecting Knowledge Compilation Classes and Width Parameters. *Theory Comput. Syst.*, *64*(5), 861–914.

Atserias, A., Bonacina, I., de Rezende, S. F., Lauria, M., Nordström, J., & Razborov, A. A. (2018). Clique is Hard on Average for Regular Resolution. In Diakonikolas, I., Kempe, D., & Henzinger, M. (Eds.), *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pp. 866–877. ACM.

Beame, P., Beck, C., & Impagliazzo, R. (2012). Time-Space Tradeoffs in Resolution: Superpolynomial Lower Bounds for Superlinear Space. In Karloff, H. J., & Pitassi, T. (Eds.), *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pp. 213–232. ACM.

Beck, C., & Impagliazzo, R. (2013). Strong ETH holds for Regular Resolution. In Boneh, D., Roughgarden, T., & Feigenbaum, J. (Eds.), *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pp. 487–494. ACM.

Ben-Sasson, E. (2002). Hard Examples for the Bounded Depth Frege Proof System. *Comput. Complex.*, *11*(3-4), 109–136.

Bodlaender, H. L., & Koster, A. M. C. A. (2006). Safe Separators for Treewidth. *Discret. Math.*, *306*(3), 337–350.

Bova, S., Capelli, F., Mengel, S., & Slivovsky, F. (2016). Knowledge Compilation Meets Communication Complexity. In Kambhampati, S. (Ed.), *Proceedings of the Twenty-*

Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp. 1008–1014. IJCAI/AAAI Press.

Buss, S., & Nordström, J. (2021). Proof Complexity and SAT Solving. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 233 – 350.

Darwiche, A. (2001). Decomposable Negation Normal Form. *J. ACM*, *48*(4), 608–647.

Darwiche, A., & Marquis, P. (2002). A Knowledge Compilation Map. *J. Artif. Intell. Res.*, *17*, 229–264.

Davis, M., Logemann, G., & Loveland, D. W. (1962). A Machine Program for Theorem-Proving. *Commun. ACM*, *5*(7), 394–397.

Davis, M., & Putnam, H. (1960). A Computing Procedure for Quantification Theory. *J. ACM*, *7*(3), 201–215.

Galesi, N., Itsykson, D., Riazanov, A., & Sofronova, A. (2019). Bounded-Depth Frege Complexity of Tseitin Formulas for All Graphs. In Rossmanith, P., Heggernes, P., & Katoen, J. (Eds.), *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, Vol. 138 of *LIPIcs*, pp. 49:1–49:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Galesi, N., Talebanfard, N., & Torán, J. (2020). Cops-Robber Games and the Resolution of Tseitin Formulas. *ACM Trans. Comput. Theory*, *12*(2), 9:1–9:22.

Glinskih, L., & Itsykson, D. (2017). Satisfiable Tseitin Formulas Are Hard for Nondeterministic Read-Once Branching Programs. In Larsen, K. G., Bodlaender, H. L., & Raskin, J. (Eds.), *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, Vol. 83 of *LIPIcs*, pp. 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Goerdt, A. (1993). Regular Resolution Versus Unrestricted Resolution. *SIAM J. Comput.*, *22*(4), 661–683.

Harvey, D. J., & Wood, D. R. (2017). Parameters Tied to Treewidth. *J. Graph Theory*, *84*(4), 364–385.

Itsykson, D., & Oparin, V. (2013). Graph Expansion, Tseitin Formulas and Resolution Proofs for CSP. In Bulatov, A. A., & Shur, A. M. (Eds.), *Computer Science - Theory and Applications - 8th International Computer Science Symposium in Russia, CSR 2013, Ekaterinburg, Russia, June 25-29, 2013. Proceedings*, Vol. 7913 of *Lecture Notes in Computer Science*, pp. 162–173. Springer.

Itsykson, D., Riazanov, A., Sagunov, D., & Smirnov, P. (2021). Near-Optimal Lower Bounds on Regular Resolution Refutations of Tseitin Formulas for All Constant-Degree Graphs. *Comput. Complex.*, *30*(2), 13.

Krajícek, J. (1995). *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, Vol. 60 of *Encyclopedia of mathematics and its applications*. Cambridge University Press.

Nordstrand, J. A. (2017). *Exploring Graph Parameters Similar to Tree-Width and Path-Width*. Ph.D. thesis, University of Bergen, Department of Informatics.

Nordström, J. (2015). On the Interplay between Proof Complexity and SAT Solving. *ACM SIGLOG News*, *2*(3), 19–44.

Razgon, I. (2016). On the Read-Once Property of Branching Programs and CNFs of Bounded Treewidth. *Algorithmica*, *75*(2), 277–294.

Scheffler, P. (1989). *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*. Ph.D. thesis.

Tseitin, G. (1968). On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, *Part 2*, 115–125.

Urquhart, A. (1987). Hard Examples for Resolution. *J. ACM*, *34*(1), 209–219.

Urquhart, A. (2011). A Near-Optimal Separation of Regular and General Resolution. *SIAM J. Comput.*, *40*(1), 107–121.

Vinyals, M., Elffers, J., Johannsen, J., & Nordström, J. (2020). Simplified and Improved Separations Between Regular and General Resolution by Lifting. In Pulina, L., & Seidl, M. (Eds.), *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, Vol. 12178 of *Lecture Notes in Computer Science*, pp. 182–200. Springer.