Electronic Colloquium on Computational Complexity, Report No. 72 (2013)

Exponential Separations in a Hierarchy of Clause Learning Proof Systems

Jan Johannsen
Institut für Informatik
Ludwig-Maximilians-Universität München
jan.johannsen@ifi.lmu.de

Abstract

Resolution trees with lemmas (RTL) are a resolution-based propositional proof system that is related to the DPLL algorithm with clause learning. Its fragments $\mathrm{RTL}(k)$ are related to clause learning algorithms where the width of learned clauses is bounded by k.

For every k up to $O(\log n)$, an exponential separation between the proof systems $\mathrm{RTL}(k)$ and $\mathrm{RTL}(k+1)$ is shown.

1 Introduction

In the past decades, a considerable amount of research has been devoted to algorithms and lower bounds for the satisfiability problem for classical propositional logic (SAT). Besides its central role in computational complexity theory, programs for this problem, so-called SAT solvers, are of increasing importance for practical applications in many domains.

Many of the most efficient contemporary SAT solvers belong to the class of conflict-driven clause learning (CDCL) solvers. Historically, these solvers developed as extensions of the basic backtracking procedure known as the DPLL algorithm [9, 8], even though their most recent versions use more general forms of backtracking.

This recursive DPLL procedure is called for a formula F in conjunctive normal form and a partial assignment α (which is empty in the outermost call). If α satisfies F, then it is returned, and if α causes a conflict, i.e., falsifies a clause in F, then the call fails. Otherwise a variable x not set by α is chosen, and the procedure is called recursively twice, with α extended by $x := 1$ and by $x := 0$. If one of the recursive calls returns a satisfying assignment, then it is returned, otherwise the call fails.

The first generations of CDCL solvers employed several refinements and

ISSN 1433-8092

extensions of the basic DPLL algorithm, including *clause learning* [14], *non-chronological backtracking* [1] and *restarts* [10]. Crucial for their success is the technique of *clause learning* [14]: when the procedure finds a conflict, a sub-assignment α' of the current assignment α is computed such that α' suffices to cause this conflict. This sub-assignment α', the reason for the conflict, can then be stored in form of a new clause added to the formula, viz. the unique largest clause $C_{\alpha'}$ falsified by α'. This way, when in a later branch of the search another partial assignment extending α' occurs, earlier backtracking is possible since then the added clause $C_{\alpha'}$ causes a conflict.

When clause learning is implemented, a strategy is needed to decide which learnable clauses to keep in memory, because learning too many clauses leads to excessive memory consumption. Early learning strategies were such that the *width*, i.e., the number of literals, of learned clauses was restricted (see e.g. [14, Sec. 3.2]). Experience has shown that such learning strategies are not very helpful, i.e., learning only short clauses does not significantly improve the performance of a DPLL algorithm for hard formulas. This experience is supported by several lower bound theorems.

Contemporary CDCL solvers use more general forms of backtracking, which are not represented by the recursive DPLL algorithm scheme above, since it may be the case that branches in the search tree pruned in backtracking contain satisfying assignments. Therefore, in the following we speak about DPLL algorithms with clause learning instead of CDCL algorithms, to make it clear that our results apply to these earlier class of algorithms, where it is enforced that no satisfying branches are pruned. It remains to be investigated whether the results carry over to more contemporary CDCL algorithms.

The first lower bound for width-restricted clause learning was shown [6] for the well-known pigeonhole principle clauses PHP_n. These formulas require time $2^{\Omega(n \log n)}$ to solve when learning clauses of width up to $n/2$ only, whereas they can be solved in time $2^{O(n)}$ when learning arbitrary clauses. Another lower bound was shown [13] for a a set of clauses Ord_n based on the principle that every finite ordering has a maximal element. These formulas can be solved in polynomial time when learning arbitrary clauses, but require exponential time to solve when learning clauses of size up to $n/4$ only. This lower bound was generalized [4] to a lower bound exponential in w for all formulas for which a lower bound w on the width of resolution refutations holds.

All these lower bounds are shown by proving the same lower bounds on the length of refutations in a certain resolution based propositional proof system RTL. The relationship of this proof system to the DPLL algorithm with clause learning was established in several earlier works [6, 12]. The learned clauses correspond to so-called *lemmas* in the proof systems, so the mentioned lower bounds were shown for a restricted version $\mathrm{RTL}(k)$ of RTL which allows only lemmas of width k, for the respective values of k.

In this work, we show that the restricted systems $\mathrm{RTL}(k)$ form a strict hierarchy: for every k, we prove an exponential separation of $\mathrm{RTL}(k+1)$ from $\mathrm{RTL}(k)$. In other words, increasing the width of lemmas that can be used by one can give an exponential speed-up.

2 Preliminaries

A *literal* is a variable x or a negated variable $\bar{x}$. A *clause* is a disjunction $C = a_1 \vee \ldots \vee a_k$ of literals a_i. The *width* of C is k, the number of literals in C. We identify a clause with the set of literals occurring in it, even though for clarity we still write it as a disjunction.

A formula in *conjunctive normal form* (CNF) is a conjunction $F = C_1 \wedge \ldots \wedge C_m$ of clauses, it is usually identified with the set of clauses $\{C_1, \ldots, C_m\}$. A formula F in CNF is in k-CNF if every clause C in F is of width $w(C) \leq k$.

We consider resolution-based refutation systems for formulas in CNF, which are strongly related to DPLL algorithms. These proof systems have as their only inference rule the *resolution rule*, which allows to infer the clause $C \vee D$ from the two clauses $C \vee x$ and $D \vee \bar{x}$, provided that the variable x does not occur in either C or D, pictorially:

$$\frac{C \vee x \qquad D \vee \bar{x}}{C \vee D}$$

We say that the variable x is *eliminated* in this inference.

A more general form of resolution inference is *w-resolution*, which allows to perform the inference even if the eliminated variable does not occur in one (or both) of the premises. More precisely, let C and D be clauses such C does not contain $\bar{x}$ and D does not contain x, then the w-resolution inference eliminating x allows to infer the clause $(C \setminus \{x\}) \cup (D \setminus \{\bar{x}\})$ from these.

The w-resolution rule can be simulated by the usual resolution rule together with the rule of *weakening* – which allows to conclude from a clause C any super-clause $D \supseteq C$ – as follows: infer $C \vee x$ and $D \vee \bar{x}$ by (possibly empty) weakenings, then apply resolution.

An *ordered binary tree* is a rooted tree in which every inner node has two children, a distinguished *left* and *right* child. The post-ordering $\prec$ of an ordered binary tree is the order in which its nodes are visited by a post-order traversal, i.e., $u \prec v$ holds for nodes u, v if u is a descendant of v, or if there is a common ancestor w of u and v such that u is a descendant of the left child of w and v is a descendant of the right child of w.

An RTL-derivation of a clause C from a CNF-formula F is an ordered binary tree, in which every node v is labeled with a clause C_v such that:

1. The root is labeled with C.

2. If v is an inner node with children u_1, u_2, then C_v follows from C_{u_1} and C_{u_2} by the resolution rule.

3. A leaf v is labeled by a clause D in F, or by a clause C labeling some node $u \prec v$. In the latter case we call C a *lemma*.

An RTL-derivation is an RTL(k)-derivation if every lemma C is of width $w(C) \leq k$. An RTL-refutation of F is an RTL-derivation of the empty clause from F.

A tree-like resolution derivation is an RTL-derivation that does not use any lemmas. An RTL-derivation is called *regular* if on every path, no variable is eliminated twice. This condition is inessential for tree-like resolution since minimal tree-like refutations are always regular [15]. It is not known whether RTL-refutations can be simulated by regular RTL-refutations without increasing the size super-polynomially.

Let V be a set of variables. A *restriction* ρ of V is a partial assignment $V \to \{0, 1\}$. A restriction ρ is extended to literals by setting

$$\rho(\bar{x}) := \begin{cases} 1 & \text{if } \rho(x) = 0 \\ 0 & \text{if } \rho(x) = 1 \end{cases}$$

For a clause C in variables V, we define

$$C\lceil\rho := \begin{cases} 1 & \text{if } \rho(a) = 1 \text{ for some } a \in C \\ \displaystyle\bigvee_{a \in C,\, \rho(a) \neq 0} a & \text{otherwise,} \end{cases}$$

where the empty disjunction is identified with the constant 0. For a CNF-formula F over V, we define

$$F\lceil\rho := \begin{cases} 0 & \text{if } C\lceil\rho = 0 \text{ for some } C \in F \\ \displaystyle\bigwedge_{C \in F,\, C\lceil\rho \neq 1} C\lceil\rho & \text{otherwise,} \end{cases}$$

where the empty conjunction is identified with 1.

Proposition 1. *Let R be a tree-like resolution derivation of C from F of size s, and ρ a restriction. Then there is a tree-like resolution derivation R' of $C\lceil\rho$ from $F\lceil\rho$ of size at most s.*

In particular, if $C\lceil\rho = 0$ then R' is a tree-like resolution refutation of $F\lceil\rho$. As usual, we denote the derivation R' by $R\lceil\rho$.

Tree-like resolution exactly corresponds to the DPLL algorithm by the following well-known correspondence: the search tree produced by the run of a DPLL algorithm on an unsatisfiable formula F forms a tree-like resolution refutation of F, and from a given tree-like regular resolution refutation of F one can construct a run of a DPLL algorithm showing the unsatisfiability of F that produces essentially the given search tree.

Buss et al. [6] define a variant WRTI of RTL which exactly corresponds to a general formulation of the DPLL algorithm with clause learning. Proofs in WRTI are regular resolution trees with lemmas using the w-resolution rule, but in which a clause can only be used as a lemma if it was derived by *input resolution*. An input resolution derivation is one in which in every inference step, one of the children is a leaf, i.e., labeled by a clause from the input formula or a lemma derived earlier.

The size of a refutation of an unsatisfiable formula F in WRTI has been shown [6] to be polynomially related to the run-time of a schematic algorithm DLL-L-UP on F. This schema DLL-L-UP subsumes many clause learning strategies commonly used in practice [6]. It follows from these results that if an unsatisfiable formula F can be solved by a DPLL algorithm with clause learning in time t, then it has an WRTI-refutation, and hence an RTL-refutation of size polynomial in t. Moreover, if the algorithm learns only clauses of width at most k, then the refutation is in RTL(k).

3 The Result

Our main result is an exponential separation between the systems RTL with lemmas restricted to be of width k, for every k:

Theorem 2. *For every k, there is a family of formulas $F_n^{(k)}$ such that*

- $F_n^{(k)}$ *have* RTL$(k+1)$*-refutations of polynomial size* $n^{O(1)}$.
- $F_n^{(k)}$ *requires* RTL(k)*-refutations of exponential size* $2^{\Omega(n/\log n)}$,

This even holds for $k = k(n)$ depending on n when $k(n) = O(\log n)$.

The lower bound also holds for a stronger system that also includes a weakening rule, the proof requires little to no modification. Therefore, it also applies for the systems with the w-resolution rule of Buss et al. [6].

On the other hand, the upper bound is shown for the weaker system with only the usual resolution rule, and the refutations given are regular.

4 Graph Pebbling

Let $G = (V, E)$ be a *pointed dag*, i.e., a directed acyclic graph having exactly one sink t, such that every vertex has either in-degree 0 or 2, and let $S, T \subseteq V$. The pebble game on (G, S, T) is played by placing pebbles onto the vertices of G according to the rules below until a pebble is placed onto a vertex in T.

Formally, a pebbling of (G, S, T) is a sequence $C_0, C_1, \ldots, C_\ell$ of subsets $C_i \subseteq V$, where C_i should be pictured as the set of vertices carrying pebbles at time i, with $C_0 = \emptyset$ and $C_\ell \cap T \neq \emptyset$ such that for all $i < \ell$ one of the following properties holds:

1. $C_{i+1} = C_i \cup \{u\}$ for some $u \in S$, i.e., a pebble can be put onto a vertex in S.
2. $C_{i+1} = C_i \cup \{u\}$ for some u such that all immediate predecessors of u are in C_i, i.e., if all predecessors of u are pebbled, then u can be pebbled.
3. $C_{i+1} \subset C_i$, i.e., pebbles can be removed from vertices.

By (2), a source vertex can be pebbled at any time, so we can always assume that S contains all sources of G.

The complexity of a pebbling is $\max_{i \le \ell} |C_i|$, i.e., the maximal number of pebbles used. The *pebbling number* $\text{peb}(G, S, T)$ is the minimal complexity of a pebbling of (G, S, T). The pebbling number $\text{peb}(G)$ of G is $\text{peb}(G, \emptyset, \{t\})$.

We shall need the following well-known property of the pebbling number [3].

Lemma 3. *For every pointed dag* $G = (V, E)$*, disjoint subsets* $S, T \subseteq V$ *and* $v \in V \setminus S \cup T$*, we have* $\text{peb}(G, S, T) \le \max(\text{peb}(G, S \cup \{v\}, T), \text{peb}(G, S, T \cup \{v\})) + 1$.

Graphs with a maximally large pebbling number were constructed by Celoni et al. [7]:

Theorem 4. *There are pointed graphs* G_n *with* n *vertices such that* $\text{peb}(G_n) \ge \Omega(n/\log n)$.

5 Pebbling Formulas

For a pointed dag $G = (V, E)$, the pebbling formula $\text{Peb}(G)$ is the unsatisfiable formula in variables x_v for $v \in V$ consisting of the following clauses:

- x_s for every source s
- $\bar{x}_u \vee \bar{x}_v \vee x_w$ for every inner vertex w with predecessors u and v
- $\bar{x}_t$ for the sink t

The formula $\text{Peb}(G)$ has a short tree-like resolution refutation of linear size, since it is a Horn formula. Ben-Sasson et al. [3] construct harder to refute formulas from them by replacing every variable x with the disjunction of two new variables $x_1 \vee x_2$. For the resulting formulas $\text{Peb}^2(G)$ they show a lower bound for tree-like resolution that is exponential in the pebbling number $\text{peb}(G)$.

6 Generalized Xorification

A different way to make a boolean formula harder is *xorification*, i.e., replacing every variable by the XOR of two or more variables. This technique has been used in proof complexity so far mainly for space lower bounds [2, 5]. It also has been applied in circuit complexity, e.g. to obtain cubic lower bounds on formula size[1] [11].

The formulas that witness the separations in Theorem 2 are obtained by xorification from the pebbling formulas $\mathrm{Peb}(G)$. In the lower bound argument, restrictions will be applied to these formulas, and in order to understand and analyze the restricted formulas, we introduce a generalized form of xorification.

We generalize xorification in two ways: first, some variables are replaced by the XOR of k variables, whereas some other variables are replaced by the negation of the XOR of k variables. Second, some designated variables are not replaced at all, but remain a single variable or its negation. Thus, for every variable two bits β_0 and β_1 specify how it occurs in the xorification: β_0 controls whether it is replaced by an XOR or not, and β_1 specifies whether it is negated or not. This is made precise in the following definition:

Let F be a formula in variables from a set V. Recall that $\neg x$ is equivalent to $x \oplus 1$. For $k \in \mathbb{N}$ and a function $\beta : V \to \{0,1\}^2$, where we denote the components of β by $\beta(x) = (\beta_0(x), \beta_1(x))$, the generalized xorification $X(F, k, \beta)$ is defined by:

- $X(x, k, \beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable $x \in V$ with $\beta_0(x) = 0$.
- $X(x, k, \beta) = x_1 \oplus \beta_1(x)$ for a variable $x \in V$ with $\beta_0(x) = 1$.
- $X(\bar{x}, k, \beta) = X(x, k, \beta) \oplus 1$ for a negated variable $x \in V$.
- $X(C, k, \beta) = \bigvee_{a \in C} X(a, k, \beta)$ expanded into CNF, for a clause C.
- $X(F, k, \beta) = \bigwedge_{C \in F} X(C, k, \beta)$ for a CNF formula F.

For the pebbling formulas $\mathrm{Peb}(G)$, we use the abbreviation $\mathrm{Peb}_{\beta}^{\oplus k}(G)$ for $X(\mathrm{Peb}(G), k, \beta)$, and we omit the lower index if β is the constant function $\beta \equiv (0,0)$. More generally, for a clause C we write $C^{\oplus k}$ for $X(C, k, \beta)$ when $\beta \equiv (0,0)$. Also we abbreviate $\beta(x_v)$ by $\beta(v)$, i.e., we identify the vertices of G with the variables of $\mathrm{Peb}(G)$.

We picture the variables of $\mathrm{Peb}^{\oplus k}(G)$ as a rectangular matrix, with a column for every vertex v of G and a row for every index $1 \leq i \leq k$.

[1] I am grateful to Ryan Williams for providing this reference on `cstheory.stackexchange.com`

The following lower bound for tree-like resolution is a generalization of the result of Ben-Sasson et al. [3], the proof is an adaptation[2] of their proof.

Theorem 5. *For every pointed dag* $G = (V, E)$ *and every* $\beta : V \to \{0,1\}^2$, *tree-like resolution refutations of* $\mathrm{Peb}_\beta^{\oplus 2}(G)$ *require size* $2^{\Omega(\mathrm{peb}(G)-b)}$, *where* b *is the number of* $v \in V$ *with* $\beta_0(v) = 1$.

Proof. Let R be a tree-like resolution refutation of $\mathrm{Peb}_\beta^{\oplus 2}(G)$, we show that $|R| \geq 2^{\mathrm{peb}(G)-b-2} - 1$.

To that end, we define a sequence $C_0, C_1, \ldots, C_h$ of clauses in R, with $C_0 = 0$ and C_{i+1} one of the predecessors of C_i for every $i < h$, and C_h a leaf, i.e., an axiom from $\mathrm{Peb}_\beta^{\oplus 2}(G)$, together with an increasing sequence of restrictions $\rho_0 \subseteq \rho_1 \subseteq \ldots \subseteq \rho_h$ such that $C_i\lceil\rho_i = 0$ for every $i \leq h$, and sets $S_0, S_1, \ldots, S_h$ and $T_0, T_1, \ldots, T_h$ with $S_i \cap T_i = \emptyset$.

We let S_0 be the set of sources in G and $T_0 = \{t\}$ where t is the sink of G, and $\rho_0 = \emptyset$. Now assume C_i, ρ_i, S_i and T_i are defined, and assume that C_i is derived from $D_i \vee x$ and $D_i' \vee \bar{x}$, where x is a variable $x_{v,\epsilon}$ for $v \in V$ and $\epsilon \in \{1, 2\}$. Let $\bar{\epsilon} := 3 - \epsilon$ so that $x_{v,\bar{\epsilon}}$ is the other variable in column v.

We define $C_{i+1}, \rho_{i+1}, S_{i+1}$ and T_{i+1} by distinguishing cases, where in each case, ρ_{i+1} is obtained from ρ_i by specifying the value for the variable $x_{v,\epsilon}$.

- Case 1a: $v \in T_i$, and $\beta_0(v) = 1$ or $x_{v,\bar{\epsilon}} \notin \mathrm{dom}\,\rho_i$.
  Set $\rho_{i+1}(x_{v,\epsilon}) = \beta_1(v)$, $S_{i+1} = S_i$ and $T_{i+1} = T_i$.
- Case 1b: $v \in T_i$ and $x_{v,\bar{\epsilon}} \in \mathrm{dom}\,\rho_i$.
  Set $\rho_{i+1}(x_{v,\epsilon}) = \rho_i(x_{v,\bar{\epsilon}}) \oplus \beta_1(v)$, $S_{i+1} = S_i$ and $T_{i+1} = T_i$.
- Case 2a: $v \in S_i$, and $\beta_0(v) = 1$ or $x_{v,\bar{\epsilon}} \notin \mathrm{dom}\,\rho_i$.
  Set $\rho_{i+1}(x_{v,\epsilon}) = \beta_1(v) \oplus 1$, $S_{i+1} = S_i$ and $T_{i+1} = T_i$.
- Case 2b: $v \in S_i$ and $x_{v,\bar{\epsilon}} \in \mathrm{dom}\,\rho_i$.
  Set $\rho_{i+1}(x_{v,\epsilon}) = \rho_i(x_{v,\bar{\epsilon}}) \oplus \beta_1(v) \oplus 1$, $S_{i+1} = S_i$ and $T_{i+1} = T_i$.
- Case 3: $v \notin S_i \cup T_i$ and $\mathrm{peb}(G, S_i, T_i \cup \{v\}) = \mathrm{peb}(G, S_i, T_i)$.
  Set $\rho_{i+1}(x_{v,\epsilon}) = \beta_1(v)$, $S_{i+1} = S_i$ and $T_{i+1} = T_i \cup \{v\}$.
- Case 4a: $v \notin S_i \cup T_i$ and $\mathrm{peb}(G, S_i, T_i \cup \{v\}) < \mathrm{peb}(G, S_i, T_i)$ and $\beta_0(v) = 1$.
  Set $\rho_{i+1}(x_{v,\epsilon}) = \beta_1(v) \oplus 1$, $S_{i+1} = S_i \cup \{v\}$ and $T_{i+1} = T_i$.

In all these cases 1a - 4a, define C_{i+1} to be the parent clause of C_i that is falsified by ρ_{i+1}.

[2]Urquhart [16] claims that a lower bound for tree-like resolution refutations of $\mathrm{Peb}^{\oplus 2}(G)$ can be obtained by imitating the proof of Ben-Sasson et al. [3] “almost word for word”. We found however that it requires some subtle modifications even for the non-generalized case.

- Case 4b: $v \notin S_i \cup T_i$ and $\mathrm{peb}(G, S_i, T_i \cup \{v\}) < \mathrm{peb}(G, S_i, T_i)$ and $\beta_0(v) = 0$. Choose C_{i+1} as that parent clause of C_i s.t. the subtree rooted at C_{i+1} is the smaller among the two, and set a value of $\rho_{i+1}(x_{v,\epsilon})$ such that C_{i+1} is falsified by ρ_{i+1}. Moreover, let $S_{i+1} = S_i \cup \{v\}$ and $T_{i+1} = T_i$.

In the following, we denote $X(x_v, 2, \beta)$ by $x_v^{\oplus}$, i.e., $x_v^{\oplus} = x_{v,1} \oplus x_{v,2} \oplus \beta_1(v)$ if $\beta_0(v) = 0$ and $x_v^{\oplus} = x_{v,1} \oplus \beta_1(v)$ if $\beta_0(v) = 1$.

Claim. *If $\rho_i(x_v^{\oplus}) = 0$, then $v \in T_i$.*

If the assumption of the claim holds, then in the case $\beta_0(v) = 1$, the value of $x_{v,1}$ must have been set in case 1a or in case 3. In either case $v \in T_i$.

In the case $\beta_0(v) = 0$, the variable among $x_{v,\epsilon}$ and $x_{v,\bar{\epsilon}}$ whose value was set later, must have been set by Case 1b, and hence $v \in T_i$.

Claim. *If $\rho_i(x_v^{\oplus}) = 1$, then $v \in S_i$.*

The proof is similar to that of the previous claim.

It follows that C_h is not a clause from a target axiom $X(\bar{x}_t, k, \beta)$. If this were the case, then $\rho_h(x_t^{\oplus}) = 1$, and hence $t \in S_h$ by the claim above, whereas we have $t \in T_0 \subseteq T_h$ and $S_h \cap T_h = \emptyset$. By analogous reasoning, C_h cannot be a clause from a source axiom $X(x_s, k, \beta)$ for a source s of G.

For $i \leq h$, let b_i be the number of $v \in V$ with $\beta_0(v) = 1$ such that $x_{v,1} \in \operatorname{dom} \rho_i$.

Claim. *For every $i \leq h$, the size s_i of the subtree of R rooted at C_i is at least $s_i \geq 2^{\mathrm{peb}(G,S_i,T_i)-b_h+b_i-2} - 1$.*

The claim is proven by induction on i, downward from h to 0.

By the considerations above, C_h must be a clause from $X(\bar{x}_u \vee \bar{x}_v \vee x_w, 2, \beta)$ for some $w \in V$ with predecessors u and v. Therefore $\rho_h(x_u^{\oplus}) = \rho_h(x_v^{\oplus}) = 1$, so $u, v \in S_h$ by the claim above, and $\rho_h(x_w^{\oplus}) = 0$, and thus $w \in T_h$. We get $\mathrm{peb}(G, S_h, T_h) = 3$, and hence $s_h = 1 = 2^{\mathrm{peb}(G,S_h,T_h)-b_h+b_h-2} - 1$, which shows the induction base for $i = h$.

Assume the claimed lower bound holds for s_{i+1}. Since s_i is the size of the tree rooted at C_i, which contains the subtree rooted at C_{i+1} of size s_{i+1}, we obviously have $s_i \geq s_{i+1}$.

If C_{i+1} is defined by one of the cases 1a through 3, then $\mathrm{peb}(G, S_{i+1}, T_{i+1}) = \mathrm{peb}(G, S_i, T_i)$ and $b_{i+1} \geq b_i$, and thus

$$s_i \geq s_{i+1} \geq 2^{\mathrm{peb}(G,S_{i+1},T_{i+1})-b_h+b_{i+1}-2} - 1 \geq 2^{\mathrm{peb}(G,S_i,T_i)-b_h+b_i-2} - 1 ,$$

which shows the claim for s_i.

If C_{i+1} was defined using case 4a, then we have

$$\mathrm{peb}(G, S_{i+1}, T_{i+1}) \geq \mathrm{peb}(G, S_i, T_i) - 1$$

by Lemma 3, and $b_{i+1} = b_i + 1$, thus we get

$$s_i \geq s_{i+1} \geq 2^{\mathrm{peb}(G,S_{i+1},T_{i+1})-b_h+b_{i+1}-2} - 1 \geq 2^{\mathrm{peb}(G,S_i,T_i)-1-b_h+b_i+1-2} - 1 ,$$

which shows the claim for s_i.

If C_{i+1} was defined using case 4b, then we have

$$\mathrm{peb}(G, S_{i+1}, T_{i+1}) \geq \mathrm{peb}(G, S_i, T_i) - 1$$

by Lemma 3 again, and $b_{i+1} = b_i$, therefore we obtain

$$s_i \geq 2s_{i+1} + 1 \geq 2^{\mathrm{peb}(G,S_{i+1},T_{i+1})-b_h+b_{i+1}-2} - 1 \geq 2^{\mathrm{peb}(G,S_i,T_i)-b_h+b_i-2} - 1 ,$$

which shows the claim for s_i.

The theorem follows, since we have $|R| = s_0$, and $b_0 = 0$, and $b_h \leq b$, and $\mathrm{peb}(G, S_0, T_0) = \mathrm{peb}(G)$. □ □

7 The Lower Bound

We will now prove one half of our main result, a lower bound on the size of RTL(k)-refutation of the $(k + 1)$-fold xorification of the pebbling formulas.

Theorem 6. *For a pointed dag* G*, every* RTL(k)*-refutation of* $\mathrm{Peb}^{\oplus(k+1)}(G)$ *requires size* $2^{\Omega(\mathrm{peb}(G))}$.

Proof. Let R be an RTL(k)-refutation of $F := \mathrm{Peb}^{\oplus(k+1)}(G)$. Note that every clause in F is of width at least k+1. Let C be the first clause in R with $w(C) \leq k$, so that C could possibly be used as a lemma. Then the subtree R_C of R rooted at C is a tree-like resolution derivation of C from F.

Let ρ be the smallest restriction with $C\lceil\rho = 0$, and note that $|\rho| \leq k$. Recall that we picture the variables of $\mathrm{Peb}^{\oplus k}(G)$ as arranged in a rectangular matrix, with a column for every vertex v of G and a row for every index $1 \leq i \leq k$. There are two cases: either the variables set by ρ are all in the same column, or ρ sets variables from at least two different columns.

In the latter case, there are at most $k-1$ rows set in every column, thus for each column v there are two rows $i(v)$ and $i'(v)$ such that $x_{v,i(v)}$ and $x_{v,i'(v)}$ are not set by ρ. In this case, we can set all but these two rows in every column, i.e., extend ρ to a restriction ρ^* by setting $\rho^*(x_{v,j}) = 0$ for every variable $x_{v,j} \notin \mathrm{dom}\,\rho$ with $j \notin \{i(v), i'(v)\}$. Define $\beta_0(v) = 0$ for every $v \in V$, and $\beta_1(v) := \bigoplus_{j\notin\{i(v),i'(v)\}} \rho^*(x_{v,j})$.

In the first case, let v be the column containing all variables set by ρ. If there are fewer than k variables set, then we can proceed as in the other case. Otherwise, there is one row i such that $x_{v,j}$ is set by ρ for all $j \neq i$. In this case, we set all but two rows in every other column, and in column v only one variable remains. Thus we pick a row i' with $i' \neq i$ arbitrarily, and extend ρ to a restriction ρ^* by setting $\rho^*(x_{u,j}) = 0$ for every column $u \neq v$ and row $j \notin \{i, i'\}$. Set $\beta_0(v) = 1$ and $\beta_1(v) = \bigoplus_{j \neq i} \rho(x_{v,j})$ for the vertex v, and for all other vertices $u \neq v$, set $\beta_0(u) = 0$ and $\beta_1(u) = 0$.

In both cases, for the so defined function β we have $F\lceil\rho^* \equiv \mathrm{Peb}^{\oplus 2}_{\beta}(G)$ after a renaming of the variables that changes only the numbering of the rows.

Thus in both cases $R_C\lceil\rho^*$ is a tree-like resolution refutation of $\mathrm{Peb}^{\oplus 2}_{\beta}(G)$, and the number b of $v \in V$ with $\beta_0(v) = 1$ is at most 1, therefore $|R_C| \geq 2^{\Omega(\mathrm{peb}(G))}$ by Theorem 5. The size lower bound for R follows. □ □

8 The Upper Bound

We now prove the remaining half of our result, the upper bound.

Theorem 7. *For every pointed dag* G *with* n *vertices, the formulas* $\mathrm{Peb}^{\oplus k}(G)$ *have regular* $\mathrm{RTL}(k)$*-refutations of size* $O(2^{3k}n)$.

Proof. Fix a topological ordering $\prec$ of G, and let S be the set of sources of G. We first show the following claim:

Claim. *Let* $w \in V$ *with predecessors* u *and* v, *where* $u \prec v$. *For every clause* C *in* $x_w^{\oplus k}$, *there is a tree-like regular resolution derivation of* C *from* $x_u^{\oplus k}$ *and* $x_v^{\oplus k}$ *and* $(\bar{x}_u \vee \bar{x}_v \vee x_w)^{\oplus k}$ *of size* $O(2^{2k})$. *Moreover, in this derivation only the variables from columns* u *and* v *are eliminated, and on every path from a leaf labeled with a clause from* $x_v^{\oplus k}$ *to* C, *only the variables from column* v *are eliminated.*

Proof. Take a regular tree-like resolution refutation R_v of $x_v^{\oplus k}$ and $\bar{x}_v^{\oplus k}$, of size $O(2^k)$. Add the clause C to every clause in R_v except the leaves from $x_v^{\oplus k}$. This yields a derivation R_v' of C from $x_v^{\oplus k}$ and $\bar{x}_v^{\oplus k} \vee C$.

Now take a regular tree-like resolution refutation R_u of $x_u^{\oplus k}$ and $\bar{x}_u^{\oplus k}$, of size $O(2^k)$. For every clause C' in $\bar{x}_v^{\oplus k} \vee C$, take a copy of R_u, and add C' to every clause in it except the leaves from $x_u^{\oplus k}$. Replace the leaf in R_v' labeled C' by the result. This gives the desired derivation and thus proves the claim. □ □

To prove the theorem, we construct a sequence $R_1, \ldots, R_\ell$ of *partial* resolution trees with lemmas, in which some leaves are labeled by clauses that are not axioms or lemmas derived earlier, these are called the *open leaves*. In addition, we define a sequence $U_1, \ldots, U_\ell$ of subsets of $V \setminus S$, such that the following invariants hold:

- The open leaves in R_i are all among the clauses from $x_{u,1} \oplus \ldots \oplus x_{u,k}$ for an $u \in U_i$.

- On the path from an open leaf with a clause from $x_{u,1} \oplus \ldots \oplus x_{u,k}$ to the root, all variables resolved are from a column $v \in V$ with $u \preceq v$.

Let R_1 be a tree-like regular resolution refutation of the clauses from $\bar{x}_t^{\oplus k}$, which are axioms of $\mathrm{Peb}^{\oplus k}(G)$, and those from $x_t^{\oplus k}$, which are the open leaves of R_1, and let $U_1 := \{t\}$. Obviously the invariants hold, and the size of R_1 is 2^k.

Assume we have constructed R_i, we show how to construct R_{i+1}. Let v be the maximal element of U_i w.r.t. the ordering $\prec$, and let u_1 and u_2 be its predecessors. For each clause C from $x_v^{\oplus k}$, replace its first occurrence in R_i by the derivation R_C of C from $x_{u_1}^{\oplus k}$ and $x_{u_2}^{\oplus k}$ given by the claim above. The other occurrences of C will then become lemmas.

Let the result be R_{i+1}, then the open leaves of R_{i+1} are those of R_i without the clauses from $x_v^{\oplus k}$, plus those leaves of R_C labeled by clauses from $x_{u_1}^{\oplus k}$ and $x_{u_2}^{\oplus k}$, except when u_1 or u_2 are sources. Thus if we define $U_{i+1} := (U_i \setminus \{v\}) \cup (\{u_1, u_2\} \setminus S)$, then the first invariant holds.

Since in a path from an open leaf C of R_i to the root, only variables from columns w with $v \preceq w$ are eliminated, and in R_C only variables from columns u_1 and u_2, the second invariant as well as regularity of R_{i+1} hold.

For each of the 2^{k-1} clauses in $x_v^{\oplus k}$, we have added one derivation of size 2^{2k}, hence the size of R_{i+1} is $|R_{i+1}| \leq |R_i| + 2^{3k}$.

The process terminates after at most n iterations, since $\max_{\prec} U_i$ strictly decreases in every step. Since in R_ℓ, there are no open leaves left, it is a regular RTL-refutation. Since every lemma used is a clause from $x_v^{\oplus k}$ for some $v \in V$, they are of size k, hence R_ℓ is a regular RTL(k)-refutation of $\mathrm{Peb}^{\oplus k}(G)$ of size $2^{3k} \cdot n$. □ □

9 Wrapping Up

Finally, we put everything together to prove the main theorem.

of Theorem 2. Let $F_n^{(k)}$ be the formula $\mathrm{Peb}^{\oplus(k+1)}(G_n)$, where G_n are the graphs given by Theorem 4 with n vertices and pebbling number $\mathrm{peb}(G_n) = \Omega(n/\log n)$. For $k = O(\log n)$, these formulas are of polynomial size $n^{O(1)}$.

By Theorem 6, the formulas $F_n^{(k)}$ require RTL(k)-refutations of exponential size $2^{\Omega(n/\log n)}$, and by Theorem 7, they have regular RTL(k)-refutations of polynomial size $n^{O(1)}$. □ □

Note that for k larger than $O(\log n)$, the formulas $\mathrm{Peb}^{\oplus(k+1)}(G_n)$ are themselves of super-polynomial size in n, and therefore have no proofs of size polynomial in the size of the underlying graph G_n.

10 Conclusion

We have shown that for resolution trees with lemmas – a resolution-based propositional proof system that forms the basis of a family of proof systems capturing the complexity of clause-learning algorithms – an increase of one in the width of clauses that may be used as lemmas can lead to an exponential speed-up.

The lower bounds hold for the strongest form of these proof systems with no regularity restrictions, and even with the weakening rule. The upper bounds, on the other hand, are given for a rather weak variant, the given refutations are regular and do not use any weakenings.

Unfortunately, we cannot immediately conclude an exponential speed-up of the DPLL algorithm with clause learning with learned clauses of width $k + 1$ over the version with learned clauses of width k from this. In order for that conclusion to hold, the upper bound would have to be given for a still weaker variant of the system, in which only lemmas derived by input resolution can be used, i.e. a restricted version of WRTI without use of the w-resolution rule and with lemmas restricted to width $k + 1$.

References

[1] Roberto J. Bayardo Jr. and Robert C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proc. 14th Natl. Conference on Artificial Intelligence*, pages 203–208, 1997.

[2] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, 2009.

[3] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near-optimal separation of general and tree-like resolution. *Combinatorica*, 24(4):585–604, 2004.

[4] Eli Ben-Sasson and Jan Johannsen. Lower bounds for width-restricted clause learning on small width formulas. In Ofer Strichman and Stefan Szeider, editors, *Theory and Practice of Satisfiability Testing – SAT 2010*, pages 16–29. Springer LNCS 6175, 2010.

[5] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011*, pages 401–416. Tsinghua University Press, 2011.

[6] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL algorithms with clause learning. *Logical Methods in Computer Science*, 4(4), 2008.

[7] James Celoni, Wolfgang Paul, and Robert Tarjan. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[8] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[9] Martin Davis and Hillary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7(3):201–215, 1960.

[10] Carla P. Gomes, Bart Selman, and Nuno Crato. Heavy-tailed distributions in combinatorial search. In Gert Smolka, editor, *Principles and Practice of Constraint Programming - CP97*. Springer LNCS 1330, 1997.

[11] Johan Håstad. The shrinkage exponent of De Morgan formulas is 2. *SIAM Journal on Computing*, 27(1):48–64, 1998.

[12] Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen van Gelder. Clause learning can effectively p-simulate general propositional resolution. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 283–290. AAAI Press, 2008.

[13] Jan Johannsen. An exponential lower bound for width-restricted clause learning. In Oliver Kullmann, editor, *Theory and Practice of Satisfiability Testing – SAT 2009*, pages 128–140. Springer LNCS 5584, 2009.

[14] João P. Marques-Silva and Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 220–227, 1996.

[15] G.S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125, 1968.

[16] Alasdair Urquhart. A near-optimal separation of regular and general resolution. *SIAM Journal on Computing*, 40(1):107–121, 2011.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de