INSTITUT FÜR INFORMATIK

der Ludwig-Maximilians-Universität München

Diplomarbeit

The Complexity of Resolution Refinements and Satisfiability Algorithms

Nicolas Rachinsky

Aufgabensteller:	Martin Hofmann
Betreuer:	Jan Johannsen, Klaus Aehlig

Contents

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

(Nicolas Rachinsky)

Abstract

Resolution is one of the most widely studied proof systems for the unsatisfiability of propositional formulas. In this work we compare the relative strength of different refinements of resolution. We study tree-like, regular, ordered, negative, semantic, and linear resolution as well as regular tree-like resolution with lemmas. We summarize and prove all the known simulations and separations between these. We present a new approach to study the strength of linear resolution with respect to general resolution. Finally, we show some lower bounds on the running time of DLL-algorithms which are based on the connection between these algorithms and resolution.

Acknowledgments

I want to thank all the people who helped me finish this work with moral support, by finding spelling and grammar errors, by checking the calculations, by explaining "obvious" steps to me,.... Since I will forget at least one, I want to apologize to all I should have mentioned but have not.

I thank Brigitte Rath, Helmut Roschy, Hendrik Grallert, Jan Hoffmann, Jan Johannsen, Klaus Aehlig, Martin Geier, Sebastian Queißer, Simon Stauber, Steffen Hausmann, Stephan Packard, my parents,...

Chapter 1

Introduction and Preliminaries

1.1 Introduction

Resolution is one of the most widely studied proof systems for the unsatisfiability of propositional formulas. Since it was introduced in the 1960's by Robinson [28], several refinements, restricted variants of resolution, were developed. Resolution and these refinements are connected with (natural) proof-search algorithms. In this work we will study tree-like, regular, ordered, negative, semantic, and linear resolution as well as regular tree-like resolution with lemmas. For example, tree-like resolution is essentially the same as the DLL algorithm, the basis for most complete **SAT**-solvers. General, regular and regular tree-like resolution with lemmas are connected with certain extensions of DLL.

Although we know the relative strength of most of these, the proofs are distributed over multiple papers which often refer to other papers for important parts of the proofs. This work presents a mostly self-contained overview of these relative strengths. It also includes proofs for the simulations and separations. Some of these are new since they were not found in the available literature, although the facts proven were mentioned. Some of the proofs were simplified or corrected.

Chapter 1 gives an overview over this work and introduces the concepts and notations used throughout this work. It also contains proofs for some important or useful properties of the introduced concepts. The most important concept introduced here is the proof system resolution and its refinements.

Chapter 2 shows an exemplary lower bound for resolution and the well-known connection between the width and size of resolution proofs.

Chapter 3, the main part of this work, contains proofs for all known simulations of the resolution refinements studied in this work as well as proofs for all known separations between them. Excepted are the results for linear resolution. These are presented in Chapter 4, which is dedicated to linear resolution. This chapter also contains a new approach to study the strength of linear resolution with respect to general resolution.

Finally, Chapter 5 shows lower bounds for DLL algorithms: First on unsatisfiable formulas by presenting and proving the well-known connection between resolution and DLL. Second, two lower bounds for DLL on satisfiable formulas are presented. These are proven by using lower bounds for resolution (on unsatisfiable formulas which have to be refuted to find a satisfying assignment).

1.2 Definitions and Preliminaries

In this work we will often use the following abbreviation.

$$[n] := \{1, \dots, n\} \text{ for } n \in \mathbb{N}$$

1.2.1 Formulas

All the formulas mentioned in this work are formulas of propositional logic.

A *literal* is either a (propositional) variable v (also v^1) or a negated variable $\neg v$ (also v^0 or $\bar{v}$). The former is called a *positive* literal and the latter a *negative* literal.

A *clause* is a disjunction $a_1 \vee \dots \vee a_k$ of literals a_i. We consider clauses to be sets, i.e., the same literal cannot occur more than once in a clause and clauses that only differ in the order of their literals are identified. The number k of literals in a clause is called its *width*. A clause is (called) *tautological* iff it contains a variable both in a positive and a negative literal. A *negative* clause is a clause that contains only negative literals, and a *positive* clause is a clause that contains only positive literals. A clause that consists of only one literal is called *unit clause*.

A formula in CNF[1] is a conjunction $C_1 \wedge \dots \wedge C_m$ of clauses C_i. The *width* of a formula is the width of its widest clause. We consider formulas to be sets, too. A *pure literal* is a variable that occurs only negatively or only positively in a formula.

An *assignment* is a mapping from the variables to $\{0, 1\}$. A *total* assignment assigns every variable (that does occur) to a value, a *partial* assignment does not necessarily assign all variables to a value (so every assignment is a partial assignment). Partial assignments are also called *restrictions*.

[1] conjunctive normal form

We will write $F\lceil_\alpha$ for the value of the formula F *restricted* by the assignment α. For a literal $a = x^\varepsilon$ we define $a\lceil_\alpha$ as

$$a\lceil_\alpha := \begin{cases} 1 & \text{if } \alpha(x) = \varepsilon \\ 0 & \text{if } \alpha(x) = 1 - \varepsilon \\ a & \text{otherwise.} \end{cases}$$

For a clause $C = a_1 \vee \ldots \vee a_k$, $C\lceil_\alpha$ is defined as

$$C\lceil_\alpha := \begin{cases} 1 & \text{if any } a_i\lceil_\alpha = 1 \\ 0 & \text{if all } a_i\lceil_\alpha = 0 \\ \bigvee_{i\in I} a_i\lceil_\alpha \text{ with } I := \{i \in [k] \mid a_i\lceil_\alpha \notin \{0,1\}\} & \text{otherwise.} \end{cases}$$

Finally, $F\lceil_\alpha$ for a formula $F = C_1 \wedge \ldots \wedge C_m$ is defined as

$$F\lceil_\alpha := \begin{cases} 0 & \text{if any } C_i\lceil_\alpha = 0 \\ 1 & \text{if all } C_i\lceil_\alpha = 1 \\ \bigwedge_{i\in I} C_i\lceil_\alpha \text{ with } I := \{i \in [m] \mid C_i\lceil_\alpha \notin \{0,1\}\} & \text{otherwise.} \end{cases}$$

For a total assignment, this always yields the value 0 or 1.

Any unsatisfiable formula must contain at least one negative and one positive clause, otherwise setting all variables to true or false, respectively, would satisfy all clauses.

SAT is the set of all formulas in CNF that are satisfiable. **SAT** (or more exactly the connected decision problem) is the most well-known **NP**-complete problem.

UNSAT is the language of all formulas in CNF that are not satisfiable. Since **SAT** is **NP**-complete, **UNSAT** is co-**NP**-complete.

1.2.2 Proof Systems

Definition 1.1. *A propositional* proof system S *is a polynomial-time computable predicate S such that for all F,*

$$F \in \mathbf{UNSAT} \iff \exists p.S(p, F).$$

Cook and Reckhow [14] defined a proof system as a polynomial-time computable *surjective* mapping $f : \Sigma^* \to \mathbf{UNSAT}$, where every string is viewed as a potential proof and f maps a (potential) proof to the formula it proves. This definition is equivalent to the one above: f_S maps (F, p) to F if $S(p, F)$ is true and to some fixed unsatisfiable formula otherwise (e.g., $x \wedge \neg x$). In the other direction, $S_f(p, F)$ is true iff $f(p) = F$.

Since we want to compare different proof systems, we introduce the notion of simulation.

Definition 1.2. *A proof system S'* simulates *a proof system S iff for every formula $F \in$ **UNSAT** and every proof p of F in S there is a is a proof p' of F in S' whose size is polynomial in $|p|$. We will write $S \leq S'$ if S' simulates S.*

Two proof systems are equivalent *iff they simulate each other.*

To show that one proof system S' does not simulate another proof system S (i.e., $S \not\leq S'$), we need at least one family φ_n $n \in \mathbb{N}$ of unsatisfiable formulas such that the smallest proof in S' is of size $f(s(\varphi_n))$ for every n where $s(\varphi_n)$ is the size of the smallest proof in S and $f(.)$ is a function of superpolynomial growth. In this case, we call S *(superpolynomially) separated* from S'. We will write $S > S'$ if we have both $S \geq S'$ and $S \not\leq S'$. Note that this no total ordering, so we might have both $S \not\leq S'$ and $S' \not\leq S$. In this case, we will write $S <> S'$ to denote the *incomparability* of S and S'.

The following observation by Cook and Reckhow [14] connects the complexity of proof systems with the question whether **NP** = co-**NP** holds.

Observation 1.3. *A proof system S such that there is, for some fixed k and every formula F, a proof p with $|p| < O(|F|^k)$ implies* **UNSAT** $\in$ **NP** *and, since* **UNSAT** *is co-***NP***-complete,* **NP** $=$ *co-***NP**. *For many proof systems formulas are known with exponential lower bounds on the proof size.*

The other direction holds, too. If **NP** $=$ *co-***NP***, then* **UNSAT** $\in$ **NP***, and the run of a nondeterministic Turing-machine that decides* **UNSAT** *could be used as a proof.*

1.2.3 Resolution

In the resolution proof system, a proof (or refutation[2]) is a resolution derivation of the empty clause $\Box$ from the input formula F. A resolution derivation of a clause C (from F) is a node labeled dag R. Each node in R has in-degree 0, 1 or 2. The nodes are labeled with clauses, and C is the label of a sink. The label of each node with in-degree 0 must be a clause in F (we call these clauses *axioms*), the label of a node with in-degree 1 must be identical to the label of its predecessor, and the label of a node with in-degree 2 must be derived from the labels of its predecessors according to the *resolution rule*:

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

where x does not occur in C or D. In this case, we say that $C \vee x$ is *resolved with $D \vee \bar{x}$ on x*, $C \vee D$ is the *resolvent* of $C \vee x$ and $D \vee \bar{x}$, or x is *eliminated.*

[2]We use proof and refutation as synonyms.

The definition includes nodes with in-degree 1 to simplify some of the proofs and to simplify the definition of regular tree-like resolution with lemmas (defined on page 14). Note that this defines resolution without weakening. Weakening is considered in Section 1.2.3.1. If not explicitly mentioned, resolution means (in this work) resolution without the weakening rule.

We will write $F \vdash C$ if C derivable with resolution from F. In a slight abuse of notation we will identify clauses and the nodes they label. We may assume that the derived clause C (most of the time $\square$) is the only sink, since we can remove all clauses from which C is not reachable. The *size* of a resolution derivation is the number of the clauses it contains, its *width* is the width of its widest clause. We will write $F \vdash_k C$ if there is a resolution derivation of C from F with a width of at most k.

Theorem 1.4. *Resolution is sound, i.e., if there is a resolution proof for a formula F, then F is unsatisfiable.*

Proof. Since any assignment satisfying $C \vee x$ and $D \vee \bar{x}$ must satisfy at least one of C or D, it satisfies $C \vee D$, too. Thus $C \vee D$ can be added to the formula without changing its satisfiability. So a proof for a formula F implies that its satisfiability does not change if the empty clause is added to it, i.e., F is unsatisfiable. Therefore resolution is sound. $\square$

Sadly resolution cannot be used to prove $\mathbf{P} = \mathbf{NP}$ by Observation 1.3 on the facing page, because there are already exponential lower bounds known. Some of these are presented in Chapter 2.

There are restricted versions of resolution (aka refinements). Without any restrictions it is called *general* or *dag-like* resolution (dag) . The following list contains the refinements considered in this work.

tree-like (tree) The dag is required to form a tree, i.e., every derived clause is used at most once (if it is needed more often, it has to be derived multiple times). This is also called DLL or DPLL (after D(P)LL algorithm, introduced in Section 1.2.4).

regular (reg) Every variable is eliminated at most once on any path from a source to the sink.

ordered (ord) There must be some linear ordering such that the variables eliminated on each path are sorted according to this linear ordering. This is also called DP or Davis-Putnam resolution.

negative (neg) One of the clauses used in each resolution step has to be a negative clause. In the same way we define the dual refinement *positive* resolution.

semantic (sem) There must be some assignment that falsifies one of the clauses used in each resolution step.

linear (lin) The dag is one chain, i.e., the result of an application of the resolution rule is used in the next step. Linear proofs can be seen as lists of clauses $C_1, \ldots, C_k$, where $C_1 \in F$ and C_i for $i > 1$ is the result of resolving C_{i-1} with some C where $C \in F$ or $C = C_j$ with $j < i$.

regular tree-like resolution with lemmas (rtrl) The dag must be one tree, every path from a leaf to the root of the tree must obey the regularity[3] constraint, i.e., each variable is eliminated at most once on the path.

There might be additional edges. These edges must lead to nodes with in-degree 1, these edges are not part of the tree, and paths through these need not obey the regularity constraint. But the starting node C of such an edge must be to the left of the ending node D, i.e., C must be in the left subtree of any tree rooted in a node on the path between D and the sink. We call D a *lemma*.

In other words, the proof has to be a regular tree, but clauses derived earlier may be used as additional axioms.

We assume that there is an annotation as to which of the edges are part of the tree and which are not for regular tree-like refutation with lemmas. And for semantic resolution we assume that one assignment satisfying the constraint is given with the proof. We won't write these explicitly to avoid cluttering the notation.

Theorem 1.5. *Resolution is complete, i.e., if a formula F is unsatisfiable, then there is a resolution proof for F.*

Proof. The completeness of resolution is shown by induction on the number of variables. W.a.l.o.g. the formula does not contain pure literals, since clauses containing these can be satisfied without any effect on the other clauses (by setting the pure literals). Thus these clauses can be removed and the formula is still unsatisfiable. We also assume that there are no tautological clauses, since these can also be removed without changing the satisfiability.

An unsatisfiable formula with no variables contains $\square$, and $\square$ is the resolution proof. Given a formula F with $n > 0$ variables, select one variable v, and partition the clauses in three sets, C_1 containing clauses where v occurs positively, C_2 containing clauses where v occurs negatively, and C_3 containing the other clauses. Now all clauses from C_1 are resolved with all clauses from C_2 on v. The resulting clauses, with tautological ones omitted, form the set C'. The formula F' consisting of the clauses from C' and C_3 has at most $n-1$ variables and is still unsatisfiable. The unsatisfiability follows

[3] One can also define this without the regularity constraint, but the resulting *tree-like resolution with lemmas* is equivalent to general resolution (see Footnote 1 on page 47).

from the fact that any satisfying assignment α for F' can be extended to one satisfying F. C_3 is already satisfied by α. If both C_1 and C_2 are satisfied, we are done. Assume α does not satisfy $c_1 \in C_1$. Since α satisfies all the clauses resolved from c_1 and the clauses in C_2, α satisfies all the clauses in C_2. C_1 is satisfied by adding $v \mapsto 1$ to α.

The resulting resolution proof is ordered (on every path the variables are eliminated in the order they were selected), and because of this the proof is also regular. And thus it is also a regular resolution proof with lemmas. It can be transformed into a tree-like proof, every clause and its derivation can be duplicated for every use of the clause, resulting in a proof where every clause is used at most once.

Now we prove the completeness of negative and semantic resolution. Since every negative refutation is also a semantic refutation, it suffices to prove that negative resolution is complete. This can be proven in the following way, which is due to Sam Buss [12]. We prove that an unsatisfiable set F of clauses cannot be closed under negative resolution and not contain the empty clause. From this the completeness follows directly, since there is always a new clause derivable with negative resolution that can be added unless there is already the empty clause in the set. And since there is only a finite number of clauses with n variables, the empty clause is reached.

Assume F is unsatisfiable and closed under negative resolution. Let A be the set of negative clauses in F, A is not empty since F is unsatisfiable. Now we select a non-partial assignment α that satisfies all clauses in A and assigns the minimal number of variables to the value false, this is possible since F does not contain the empty clause. Now we take a clause C from $F \setminus A$ which is falsified by α and which has the minimal number of positive literals. Such a clause exists since F is unsatisfiable and α is a full assignment satisfying all negative clauses. Let x be one of the positive literals in C. Note that $\alpha(x) = 0$ since C is falsified by α. Now we take a clause $D \in A$ containing x as the only variable set to false, i.e., the literal $\bar{x}$ is the only satisfied literal in the negative clause D. Such a clause exists by the choice of α. If all clauses in A containing $\bar{x}$ have another variable set to false, then x would be assigned to true. And there is at least one clause in A that contains $\bar{x}$ since otherwise x would be assigned to true, too. Now consider the resolvent R of C and D. R is falsified by α, therefore it cannot be contained in A, it contains less positive literals than C, so it cannot be in $F \setminus A$. Therefore it must be a new clause, and this contradicts the assumption that F is closed under negative resolution.

Completeness of linear resolution follows from Theorem 4.1 on page 61.

□

Observation 1.6. *The above proof shows that there is an ordered refutation of a formula F for any ordering of the variables. Since every variable occurs at most once along this path, any path in this refutation has at most length n, thus the whole refutation has at most size $2^{O(n)}$.*

The following result is often useful.

Theorem 1.7. *If there is a resolution derivation R of $\widetilde{C}$ of size s and width w from an unsatisfiable formula F, then given a partial assignment α that falsifies $\widetilde{C}$, there is a resolution proof R' for $F\lceil_\alpha =: F'$ of size at most s and width at most w.*

If R is regular, negative, semantic, tree-like or ordered, then so is R'.

Proof. Let R be the resolution derivation of $\widetilde{C}$ from F of size s. We change this into a proof R' of F'. First we replace all clauses C from F in R by the clauses $C\lceil_\alpha$ from F'.

Let E be the resolvent of $C \vee p$ and $D \vee \bar{p}$, which are replaced by C' and D'. If C' contains p and D' contains $\bar{p}$, E is replaced by the resolvent of C' and D' on p. Otherwise E is replaced by C' if C' does not contain p, or by D' if C' does contain p. We denote the replacement of E by E'.

In all cases, E' does not contain p or $\bar{p}$ and is a subset of E, and E' does not contain any variables that are assigned by α. Thus we get a resolution derivation of $\widetilde{C}'$ from F', where $\widetilde{C}'$ is a subset of $\widetilde{C}$ that does not contain any variable set by α, thus it is the empty clause, and R' is a refutation of F'.

Since we do not add clauses and possibly remove some if we remove parts of the dag not leading to $\Box$, $|R'| \leq s$. Since we do not add literals to any clause, R' cannot be wider than R.

Since we just leave out applications of the resolution rule, we do not change the order of the eliminations, thus an ordered proof is transformed into an ordered one, and a regular one is transformed into a regular one. In the resulting proof, both of the clauses used in a resolution step are subsets of two clauses used in a resolution step of the original proof, thus a semantic or negative proof is transformed into a semantic or negative one. Since we do not add any new edges to the dag, a tree-like proof is transformed into a tree-like proof. $\Box$

This implies the following corollary.

Corollary 1.8. *If there is a resolution proof R of size s and width w for an unsatisfiable formula F, then given a partial assignment α there is a resolution proof R' for $F\lceil_\alpha =: F'$ of size at most s and width at most w.*

If R is regular, negative, semantic, tree-like or ordered, then so is R'.

This proof does not work for linear resolution and regular tree-like resolution with lemmas. It is still unknown if the above theorem and corollary hold for these.

Theorem 1.9. *If there is a tree-like resolution proof of size s for an unsatisfiable formula F, there is a regular tree-like resolution proof of size at most s for F.*

Proof. An irregularity can be removed in the following way.

A non-regular proof does contain a step

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

where x is eliminated again on the path $C_1 = C \vee D, \ldots, C_l = \square$. Let C_m be the first clause containing x or $\bar{x}$, w.a.l.o.g. it contains x.

Now the proof is changed in the following way. $D \vee \bar{x}$, its derivation and C_1 are removed from the proof. The clauses $C_2, \ldots, C_{m-1}$ are replaced by a clause $C'_i \vee x$ where $C'_i \subseteq C_i$. If C'_{i-1} does not contain the literal l_i eliminated from C_{i-1} and D_{i-1} in the step yielding C_i, we remove D_{i-1}, its derivation and C_i. Otherwise C'_i is the result of resolving C'_{i-1} with D_{i-1}. The clauses $C_k, \ldots, C_{m-1}$ are replaced by $C'_k, \ldots, C'_{m-1}$ in the same way (but here C'_i does not contain x unless C_i contained one).

We repeat this until there are no more irregularities. Since there are at least two clauses deleted in every step, the resulting proof is at most of the size of the original one. $\square$

Note that this proof does not work for general resolution, since clauses can be used multiple times and it might be necessary to duplicate such clauses when they are modified. In Section 3.3 we will prove that the theorem does not hold for general resolution.

1.2.3.1 Weakening

Sometimes the definition of resolution allows an additional rule, called *weakening rule*.

$$\frac{C}{C \vee x}$$

This rule is not needed for completeness, and resolution with this rule is still correct, since every assignment satisfying C satisfies the weakened clause $C \vee x$ as well, so the proof of Theorem 1.4 on page 13 still works.

The weakening rule does not change the size of the smallest proofs for most refinements of resolution.

Theorem 1.10. *If there is a resolution proof R of size s using the weakening rule, then there is a resolution proof R' of size at most s not using the weakening rule. If R is regular, negative, semantic, tree-like or ordered, then so is R'.*

Proof. We prove this by transforming a resolution refutation that uses the weakening rule into one that does not use it, without increasing its size.

First, we move every application of the weakening rule as far towards the empty clause as possible (possibly duplicating the application of the weakening rule if the variable added is used multiple times). After this step every variable added by the weakening rule is removed by resolution in the next step of the refutation.

Second, we remove all useless applications of the weakening rule, i.e., where the added literal was already present in the clause.

All remaining applications of the weakening rule are of the form

$$\dfrac{\dfrac{C}{C \vee l} \quad D \vee \bar{l}}{C \vee D}$$

We remove such an application of the weakening rule by leaving out both of these steps and replacing $C \vee D$ by C. Now all clauses C_i derived (directly and indirectly) from $C \vee D$ are replaced by $C'_i \subseteq C_i$ as necessary to keep it a resolution refutation, in the same way as in the proof of Theorem 1.7 on page 16. We repeat the last step until there is no application of the weakening rule left.

Since we added only uses of the weakening rule (in the first step) and removed all of them later, the resulting proof is at most as big as the original one. The refinements are preserved for the same reasons as in the proof of Theorem 1.7. □

This proof does not work for linear resolution and regular tree-like resolution with lemmas. It is possible that these refinements are stronger with weakening than without it.

Theorem 1.11. *If there is a resolution refutation R of a formula F, then there is a refutation R' of F' where F' is a subset of F such that every clause in $F \setminus F'$ is the superset of some clause in F'. Furthermore the size of R' is at most $|R| + \max(|F \setminus F'|, |R|) \cdot w(F)$ where $w(F)$ is the width of F.*

If R is regular, negative, semantic, tree-like or ordered, then so is R'.

Proof. Every removed clause can be derived with at most $w(F)$ weakening steps from a clause that is in F'. We need to do this once for every use of a removed clause as an axiom. There are at most $\max(|F \setminus F'|, |R|)$ such uses of an axiom.

By Theorem 1.10 on the previous page we can leave out the weakening steps without increasing the refutation any further. Therefore this proof does not work for linear resolution and regular tree-like resolution with lemmas. □

In the following chapters we will not use weakening where it can be avoided without causing too much hassle.

1.2.3.2 Tautological Clauses

Tautological clauses in a formula have a similar effect as the weakening rule. For general, regular, negative, semantic, tree-like and ordered resolution they have no influence on the size of the smallest proof.

Theorem 1.12. *If there is a resolution proof R for a formula F, then there is a proof R' for F' where F' is the subset of non-tautological clauses of F. Furthermore R' does not contain tautological clauses and $|R'| \leq O(|R| \cdot w(R))$ where $w(R)$ is the width of R.*

If R is regular, negative, semantic, tree-like or ordered, then so is R'.

Proof. We construct R' from R by removing every occurrence of a tautologic clause T in the following way. First we construct a proof that uses the weakening rule.

If the only variable occurring twice in $T = x \vee \bar{x} \vee T'$ is eliminated from T and C as soon as T occurs in R, this is simulated by adding T' with the weakening rule to C.

Otherwise, x and $\bar{x}$ are removed later by resolving with C and C' yielding D. Then C and C' can be resolved on x, and the missing literals from D are added by the weakening rule.

Then we remove the weakening rule again with Theorem 1.10 on page 17. This proves the theorem. Again this proof does not work for linear resolution and regular tree-like resolution with lemmas. □

1.2.4 DLL Algorithms

DLL algorithms (also called DPLL algorithms; named after Davis, Putnam, Logemann, and Loveland; [16] and [15]) and variations are the algorithms used most often to solve **SAT** problems.

The algorithm is called with a formula and a partial assignment. It first checks whether the formula is satisfied by the assignment or is trivially unsatisfiable, in these cases it returns the current assignment or UNSATISFIABLE, respectively. Otherwise a not yet set variable v and a truth value ε is selected by a heuristic. The algorithm adds the setting $v \mapsto \varepsilon$ to the assignment and calls itself recursively with the new assignment. If the recursive call returns an assignment, this is returned. Otherwise the setting $v \mapsto 1 - \varepsilon$ is added to the (original) assignment and the algorithm calls itself with this assignment. If the recursive call returns an assignment, this is returned. Otherwise UNSATISFIABLE is returned. Pseudocode for this algorithm is shown in Figure 1.1 on the next page.

Since these algorithms are complete, i.e., they return a correct answer SAT/UNSAT after a finite running time, a (log of a) run of a DLL algorithm returning UNSAT is a proof for the unsatisfiability of the input formula.

There are many variations of the base algorithm. It is possible to simplify the formula at the beginning of each call of DLL(). The most noteworthy

```
DLL(F, α)
    if F⌈_α = 1
        return α
    if F⌈_α = 0
        return UNSAT

    (v, ε) := HEUR(F, α)
    # HEUR is the heuristic that selects
    # the variable v to be set next and the value ε
    # for v that should be tried first.
    # v is called decision variable.

    σ := DLL(F, α ∪ {v ↦ ε})

    if σ ≠ UNSAT
        return σ
    else
        return DLL(F, α ∪ {v ↦ ¬ε})
```

Figure 1.1: DLL Algorithm

simplification is *unit propagation*, i.e., setting variables occurring in unit clauses to the one possible value as long as possible. Pseudocode is shown in Figure 1.2 .

```
UP(F, α)
    while F⌈_α contains a unit
        select a unit clause l (here l is a literal, i.e., l = v or l = v̄)
        α = α ∪ {l ↦ 1}
```

Figure 1.2: Unit Propagation

The other two most prominent simplifications are *pure literals*, i.e., all clauses that contain a pure literal can be removed, since we can set a pure literal such that all these clauses are satisfied without affecting any other clause, and *subsumption*, i.e., a clause D is removed if there is a clause C with $C \subset D$, this is correct since any assignment satisfying C does also satisfy D. Both of these are mostly of theoretical use, since they are quite slow (for some practical/empirical definition of slow).

Another important improvement is called *(clause) learning*. This adds new clauses that do not change satisfiability to the formula when the algorithm has to backtrack.

An algorithm using learning may also do *restarts*, i.e., forget the current partial assignment and start with an empty one. The learned clauses and other information gathered are kept.

Restarts must be handled with care, since they might remove the completeness, i.e., there must be some guarantee that the algorithm cannot enter an infinite loop. One easy way is to increase the number of steps before the next restart can happen after every restart.

1.2.5 Pebbling

Let $G = (V, E)$ be a directed acyclic graph (*dag*), and let $S, T \subset V$. While the following definitions work for dags whose nodes have any (finite) in-degree, we will only use them for graphs whose nodes have in-degree 0 and 2.

A *pebbling* (G, S, T) means to put pebbles on nodes of the graph according to the following rules until there is a pebble on a node in T.

1. A pebble may be put on any node in S.
2. A pebble may be removed from any node at any time.
3. A pebble may be put on a node v when there are pebbles on all direct predecessors of v.

More formally, a pebbling (G, S, T) is a sequence $C_0, C_1, \ldots, C_k$ of sets $C_i \subseteq V$, with $C_0 = \emptyset$ and $C_k \cap T \neq \emptyset$. And for all $0 \leq i < k$, one of the following properties holds:

1. $C_{i+1} = C_i \cup \{u\}$ with $u \in S$
2. $C_{i+1} \subset C_i$
3. $C_{i+1} = C_i \cup \{u\}$ if all direct predecessors of u are in C_i

Here each C_i is the set of nodes with a pebble after the i-th step.

The complexity of a pebbling is the number of pebbles needed, i.e., $\max_{i \leq k}(|C_i|)$. The *pebbling number* $\mathrm{Peb}(G, S, T)$ of (G, S, T) is the minimal complexity of the pebblings from S to T. The pebbling number $\mathrm{Peb}(G)$ of a dag G is $\mathrm{Peb}(G, S, T)$ with S the set of the sources in G and T the set of the sinks in G.

Lemma 1.13. *For G, S, T as above and $v \in V$ the following holds*

$$\mathrm{Peb}(G, S, T) \leq \max(\mathrm{Peb}(G, S, T \cup \{v\}), \mathrm{Peb}(G, S \cup \{v\}, T) + 1).$$

Proof. First we can pebble G from S to $T \cup \{v\}$ using $\mathrm{Peb}(G, S, T \cup \{v\})$ pebbles. If this ends with a pebble on a node in T we're done. Otherwise we remove all pebbles except the one on v. Now we can use this one to continue with a pebbling from $S \cup \{v\}$ to T, but we never remove the pebble on v. Thus this needs at most $\mathrm{Peb}(G, S \cup \{v\}, T) + 1$ pebbles. $\square$

There are graphs with large pebbling numbers.

Theorem 1.14 (Celoni et al. [26])**.** *There are graphs G_n with n vertices such that*

$$\mathrm{Peb}(G_n) \geq \Omega(n/\log n),$$

for large n.

Chapter 2

Lower Bounds for Resolution

In this chapter we show an exemplary lower bound for resolution and the well known connection between the width and size of tree-like and general resolution proofs.

2.1 Lower Bound for PHP

The first lower bound we will show is an exponential lower bound for resolution refutations of PHP_n^{n+1} (or more exactly, we will show the lower bound for PHP_{n-1}^n, since the terms are simpler in this case). PHP_n^m (with $m > n$) is an unsatisfiable set of clauses stating the negation of the PigeonHole Principle. The latter states that there cannot be an $1-1$ mapping from a set of cardinality m into a set of cardinality n with $m > n$. This was first proven by Haken [20]. The presented (simpler) proof is due to Beame and Pitassi [5].

PHP_n^m uses the variables $p_{i,j}$ with the intended meaning of putting pigeon i into hole j. Then PHP_n^m consists of the following clauses:

- the pigeon clauses, stating that every pigeon is put into a hole:

$$P_i = \bigvee_{j \in [n]} p_{i,j} \text{ for every } i \in [m]$$

- the hole clauses, stating that at most one pigeon is put into a hole:

$$H_{i,j,k} = \bar{p}_{i,k} \vee \bar{p}_{j,k} \text{ for every } i < j \in [m] \text{ and } k \in [n]$$

Theorem 2.1. *If P is a resolution refutation of PHP_{n-1}^n, then $|P| \geq 2^{n/20}$.*

To prove this, we first introduce (or recall) some definitions. A *matching* ρ from $[m]$ into $[n]$ is a set of pairs

$$\{(i_1, j_1), \ldots, (i_k, j_k)\} \subset [m] \times [n]$$

such that all the i_ν as well as all the j_ν are pairwise distinct. The size of ρ is $|\rho| = k$.

A matching induces a partial assignment of the variables of PHP:

$$\rho(p_{i,j}) = \begin{cases} 1 & \text{if } (i,j) \in \rho \\ 0 & \text{if there is } (i,j') \in \rho \text{ with } j' \neq j \\ & \qquad\quad \text{or } (i',j) \in \rho \text{ with } i' \neq i \\ \text{undefined} & \text{otherwise} \end{cases}$$

We identify a matching with the partial assignment it induces.

There is also a total truth assignment α_ρ induced by a matching where every unset variable is set to zero, i.e.,

$$\alpha_\rho(p_{i,j}) = \begin{cases} 1 & \text{if } (i,j) \in \rho \\ 0 & \text{otherwise} \end{cases}$$

Note that since a matching puts $|\rho|$ pigeons into distinct holes, α_ρ satisfies $|\rho|$ pigeon clauses and all the hole clauses.

A truth assignment α_ρ given by a maximal matching (i.e., $|\rho| = n$) is called *critical assignment*.

Definition 2.2. *For $I \subseteq [m]$ and $J \subseteq [n]$ we define the following abbreviation (stating one of the pigeons in I is put into one of the holes in J).*

$$P_{I,J} := \bigvee_{i \in I} \bigvee_{j \in J} p_{i,j}$$

Using this notation the pigeon clause P_i is $P_{\{i\},[n]}$.

For the proof, we use the *monotone calculus*, which was introduced by Buss and Pitassi [13]. The monotone calculus is a proof system where a proof is a list of positive (also called monotone) clauses, where, similar to resolution proofs, clauses must either be taken from the set of clauses to refute or derived via the only rule from clauses occurring earlier in the list, where the list must end in the empty clause. It was specifically designed to refute PHP clauses. The rule is

$$\frac{C \vee P_{I_0,\{j\}} \quad D \vee P_{I_1,\{j\}}}{C \vee D}$$

with $I_0, I_1 \subset [m]$ and $I_0 \cap I_1 = \emptyset$.

It is easy to see that every assignment α_ρ for a matching ρ that satisfies $C \vee P_{I_0,\{j\}}$ and $D \vee P_{I_1,\{j\}}$ satisfies $C \vee D$, thus the monotone calculus is sound in respect to such assignments. Furthermore it is equivalent to resolution on PHP formulas.

Lemma 2.3. *If there is a monotone refutation of the pigeon clauses of PHP^m_n of size s, then there is a resolution refutation of size at most $s \cdot m^2$.*

Proof. Every step in the monotone proof can be simulated by m^2 resolution inferences.

$$\frac{C \vee P_{I_0,\{j\}} \quad D \vee P_{I_1,\{j\}}}{C \vee D}$$

$D \vee \bar{p}_{i,j}$ can be derived from $D \vee P_{I_1,\{j\}}$ for every $i \in I_0$ within $|I_1|$ steps, by using the hole clauses $\bar{p}_{i,j} \vee \bar{p}_{i',j}$ with $i' \in I_1$.

Resolving these $|I_0|$ clauses with $C \vee P_{I_0,\{j\}}$ results in a derivation of $C \vee D$, using $|I_0| \cdot (|I_1| + 1)$ steps.

$$|I_0| \cdot (|I_1| + 1) \leq \frac{m}{2}(\frac{m}{2} + 1) = \frac{m^2}{4} + \frac{m}{2} \leq m^2$$

□

The above lemma is not needed to prove the lower bound for PHP, it is included here for completeness only.

Lemma 2.4. *If there is a resolution refutation R of* PHP^m_n *of size s, then there is a monotone refutation R' of the pigeon clauses of size at most s.*

Proof. We define C^+ and C^- to be the set of positive and negative literals of a clause C. Now we define a positive clause C^m for every clause C that is equivalent with C under critical assignments, i.e., $C\lceil_{\alpha_\rho} = C^m\lceil_{\alpha_\rho}$ for ρ a matching with $|\rho| = n$.

$$C^m := \bigvee_{x_{i,j} \in C^+} x_{i,j} \vee \bigvee_{x_{i,j} \in C^-} P_{[m]\setminus\{i\},\{j\}}$$

We now construct R' from R by replacing every clause C by another clause C' or removing it. Thus R' is at most of the size of R. We maintain $C' \subseteq C^m$ for every clause in R', so R' will again end with the empty clause.

If C is a pigeon axiom P_l, then $C' := C = P_l$. If C is a hole axiom $H_{l,j,k}$, then it is removed.

If $C = \widetilde{D}_0 \vee \widetilde{D}_1$ is the resolvent of $D_0 = \widetilde{D}_0 \vee p_{i,j}$ and $D_1 = \widetilde{D}_1 \vee \bar{p}_{i,j}$, then C' is either D'_0, D'_1 or it is derived by one monotone step from D'_0 and D'_1, thus R' is a monotone proof. Note that D_0 cannot be a hole clause and is therefore replaced by some clause D'_0.

- If $p_{i,j} \notin D'_0$, we set $C' := D'_0 \subseteq C^m$.
- Otherwise if D_1 is a hole clause $H_{i,k,j}$, then $p_{i,j} \in \widetilde{D}^m_1 = P_{[m]\setminus\{k\},\{j\}}$ and we set $C' := D'_0 \subseteq \widetilde{D}^m_0 \vee \widetilde{D}^m_1 = C^m$.
- Otherwise D_1 is replaced by a clause $D'_1 \subseteq D^m_1 = \widetilde{D}^m_1 \vee P_{[m]\setminus\{i\},\{j\}}$.
  - If $p_{i,j} \in D'_1$, then $p_{i,j} \in \widetilde{D}^m_1$. We set $C' := D'_0 \subseteq \widetilde{D}^m_0 \vee \widetilde{D}^m_1 = C^m$ in this case.

– Otherwise if $p_{i',j} \notin D'_1$ for every $i' \neq i$, then $D'_1 \subseteq \widetilde{D}^m_0 \subseteq C^m$, and we set $C' := \widetilde{D}'_1$.
– Otherwise $D'_1 = \widetilde{D}'_1 \vee P_{I,\{j\}}$ for some $\widetilde{D}'_1$ and I with $\widetilde{D}'_1 \subseteq \widetilde{D}^m_1$ and $i \notin I$. We set $C' := \widetilde{D}'_0 \vee \widetilde{D}'_1 \subseteq \widetilde{D}^m_0 \vee \widetilde{D}^m_1 = C^m$. Note that the same C' is obtained from D_0 and D_1 via the monotone rule. □

Because of the above lemma, it is enough to prove a lower bound for the monotone calculus to prove one for resolution. We will now show that every short monotone proof for PHP can be converted to one that contains only short clauses.

Lemma 2.5. *If R is a monotone refutation of* PHP^n_{n-1} *of size $|R| < 2^{n/20}$, there is a matching ρ with $|\rho| \leq 0.329n$ such that $R\lceil_\rho$ does not contain any clause C with at least $n^2/10$ variables.*

Proof. We will call a clause C *wide* if it contains more than $n^2/10$ variables. Note that every wide clause contains at least $\frac{1}{10}$ of all variables. Thus the probability of a randomly chosen variable to occur in some fixed wide clause is therefore at least 1/10. Thus there is a variable that occurs in at least 1/10 of the wide clauses. We now construct a matching inductively using the following greedy algorithm.

$$\begin{aligned} \rho_0 &:= \emptyset \\ \rho_{k+1} &:= \rho_k \cup \{(i,j)\} && \text{with } p_{i,j} \text{ one of the variables occurring} \\ & && \text{most often in wide clauses in } R\lceil_{\rho_k} \end{aligned}$$

Let s be the number of wide clauses in R and s_k the number of wide clauses in $R\lceil_{\rho_k}$. Then $s = s_0$ and $s_{k+1} \leq \frac{9}{10}s_k$. Therefore $s_r = 0$ for $r := \lceil \log_{10/9} s \rceil$. We define $\rho := \rho_r$.

$$|\rho| = r = \lceil \log_{10/9} s \rceil \leq \log_{10/9}(2^{n/20}) = ((log_{10/9}2)/20)n < 0.329n$$

That and the fact that $R\lceil_{\rho_r}$ does not contain any wide clause prove the lemma. □

Now we prove that every monotone proof of PHP must contain at least one wide clause.

Lemma 2.6. *If R is a monotone refutation of* PHP^n_{n-1}*, then there is a clause C in R containing at least $2n^2/9$ variables.*

Proof. We first define for F a subset[1] of PHP^n_{n-1} and C a positive clause $F \models_{cr} C$ iff every critical assignment satisfying F does satisfy C.

[1] We are only interested in pigeon clauses here, but hole clauses in F do not matter since every critical assignment satisfies all hole clauses.

Then we define the following measure on positive clauses.

$$\mu(C) := \min\{|F| \mid F \subseteq \mathrm{PHP}^n_{n-1} \text{ and } F \models_{cr} C\}$$

We have $\mu(P_i) = 1$ and $\mu(\Box) = n$. For clauses C_1, C_2, C_3 with $C_1, C_2 \models_{cr} C_3$ we have $\mu(C_3) \leq \mu(C_1) + \mu(C_2)$. Thus there must be a clause C in R with $\frac{n}{3} < \mu(C) \leq 2 \cdot \frac{n}{3}$.

Take $F \subset \mathrm{PHP}^n_{n-1}$ minimal with $F \models_{cr} C$, i.e., $\frac{n}{3} < |F| \leq 2 \cdot \frac{n}{3}$. We will now show that the clause C contains at least $|F| \cdot (n - |F|)$ variables. $|F| \cdot (n - |F|) \geq 2n^2/9$ since the function $x \mapsto x(n-x)$ has its minimum in the interval $\frac{n}{3} < x \leq 2 \cdot \frac{n}{3}$ at the endpoints where the value is $2n^2/9$.

Now let $1 \leq i \leq n$ with $P_i \in F$ and let α_ρ be a critical assignment with $\alpha_\rho \not\models P_i$ and $\alpha_\rho \not\models C$. Such an α_ρ does exist since F is minimal. Note that P_i is the only pigeon clause not satisfied by α_ρ. Now let $1 \leq j \leq n$ with $P_j \notin F$ and let k_j with $(j, k_j) \in \rho$. Now we define $\rho_j := (\rho \setminus \{(j, k_j)\}) \cup \{(i, k_j)\}$. Note that α_{ρ_j} is still a critical assignment since ρ_j is still a matching and $|\rho| = |\rho_j|$, because of $\alpha_\rho \not\models P_i$. Thus $\alpha_{\rho_j} \models F$ and since $F \models_{cr} C$, we also get $\alpha_{\rho_j} \models C$. Since p_{i,k_j} is the only variable set to 1 by α_{ρ_j} but not α_ρ and C is a positive clause, C must contain p_{i,k_j}.

This holds for every combination of $P_i \in F$ and $P_j \notin F$, thus C must contain at least $|F| \cdot (n - |F|) \geq 2n^2/9$ variables. $\Box$

Proof (of Theorem 2.1). Let R be a monotone refutation of PHP^n_{n-1} with $|R| < 2^{n/20}$. By Lemma 2.5 on the preceding page there is a matching ρ with $|\rho| < 0.329n$ and $R\lceil_\rho$ does not not contain any clause with at least $n^2/10$ variables.

But $R\lceil_\rho$ is a monotone proof for $\mathrm{PHP}^{n'}_{n'-1}$ (after renaming the variables) with $n' \geq 0.671n$. Thus by Lemma 2.6 on the facing page R must contain a clause of length $2(0.671n)^2/9 > 0.9n^2/9 = n^2/10$ variables. This is a contradiction to the above upper bound on the clause length. Thus $|R| \geq 2^{n/20}$. $\Box$

2.2 Short Proofs are Narrow

There is a a connection between the length and the width of resolution proofs. This was proven by Ben-Sasson and Widgerson [7].

In this section we will use $w(X)$ to denote the width of X, where X might be a clause, a resolution derivation or a set of clauses. In the latter cases it denotes the maximal width of a clause occurring in X. We will use $w(F \vdash C)$ to denote the width of the smallest width derivation of C from F. Here we will use the previously defined notation $F \vdash_l C$ to state there is a derivation of C from F with width l.

Lemma 2.7. *For $\varepsilon \in \{0,1\}$ and an unsatisfiable formula F, if $F\lceil_{x:=\varepsilon}\vdash_k C$, then $F \vdash_k C$ or $F \vdash_{k+1} C \vee x^{1-\varepsilon}$.*

Proof. Let R be the derivation of C from $F\lceil_{x:=\varepsilon}$ with width at most k. Let F' be the set of those clauses in F that contain $x^{1-\varepsilon}$.

We now construct R' by adding $x^{1-\varepsilon}$ to all clauses in R that come from $F'\lceil_{x:=\varepsilon}$ and all clauses derived from these (directly and indirectly).

If R does not use any clause from $F'\lceil_{x:=\varepsilon}$, $R = R'$ and R' is a derivation of C from F.

Otherwise R' is a resolution derivation of $C \vee x^{1-\varepsilon}$ from F, since $x^{1-\varepsilon}$ is not removed in R and C is derived (indirectly) from a clause containing $x^{1-\varepsilon}$. The width of the new derivation is at most $k+1$ since one literal is added to some of the clauses. $\square$

Lemma 2.8. *If $F\lceil_{x:=\varepsilon}\vdash_{k-1} \square$ and $F\lceil_{x:=1-\varepsilon}\vdash_k \square$ hold, then $F \vdash_l \square$ with $l = \max(k, w(F))$.*

Proof. Let R be a resolution derivation of $\square$ from $F\lceil_{x:=1-\varepsilon}$ with width at most k. Let F_{x^ε} be the set of all clauses of F not containing x^ε.

By Lemma 2.7 we can derive $\square$ or $x^{1-\varepsilon}$ from F with width k. In the first case we are done. In the second case, we resolve $x^{1-\varepsilon}$ with all clauses in F that contain x^ε. This has width $w(F)$. R is a derivation of $\square$ from these clauses and F_{x^ε} of width k. $\square$

Theorem 2.9.

$$w(F \vdash \square) \leq w(F) + \log s_t(F)$$

where $s_t(F)$ is the size of the smallest tree-like resolution refutation of F.

Proof. We prove by induction on (b, n) that $s_t(F) \leq 2^b$ implies $w(F \vdash \square) \leq w(F) + b$. If $b = 0$, then $\square \in F$ and we are done.

Otherwise the proof ends with

$$\frac{x \quad \bar{x}}{\square}$$

Now let R_x be the derivation of x from F and let $R_{\bar{x}}$ be the derivation of $\bar{x}$ with sizes S_x and $S_{\bar{x}}$. We know $S_x + S_{\bar{x}} + 1 \leq 2^b$. W.a.l.o.g. $S_x \leq 2^{b-1}$.

$R_x\lceil_{x:=0}$ is a derivation of $\square$ from $F\lceil_{x:=0}$ and $R_{\bar{x}}\lceil_{x:=1}$ is a derivation of $\square$ from $F\lceil_{x:=1}$.

By induction on b we know that $w(F\lceil_{x:=0}\vdash \square) \leq w(F) + b - 1$ and by induction on n we know that $w(F\lceil_{x:=1}\vdash \square) \leq w(F) + b$. With Lemma 2.8 we get $w(F \vdash \square) \leq w(F) + b$. $\square$

Corollary 2.10. *The size of a tree-like resolution refutation of a formula F is at least $2^{(w(F\vdash\square)-w(F))}$.*

Theorem 2.11.

$$w(F \vdash \Box) \le w(F) + O\left(\sqrt{n \ln s_g(F)}\right)$$

where $s_g(F)$ is the size of the smallest (general) resolution refutation of F.

Proof. If $s_g(F) = 1$ we are done. Otherwise let R be a refutation of size $s_g(F)$.

Now set $d := \left\lceil \sqrt{2n \ln s_g(F)} \right\rceil$ and $a := (1-(d/2n))^{-1}$. Let R^* denote the set of wide clauses in R, where wide means a width greater than d. We now prove by induction on (b, n) that $|R^*| < a^b$ implies $w(F \vdash \Box) \le d + w(F) + b$.

If $b = 0$, then there is no wide clause in R, and hence $w(F \vdash \Box) \le d + w(F) + b$ is true.

Otherwise one of the $2n$ literals (w.a.l.o.g. x) appears in at least $\frac{d|R^*|}{2n} = \frac{d}{2n}|R^*|$ of the clauses in R^*. Therefore there are at most $(1 - \frac{d}{2n})|R^*| < a^{b-1}$ wide clauses in $R\lceil_{x:=1}$. By induction on b we have $w(F\lceil_{x:=1} \vdash \Box) \le d + w(F) + b - 1$, and by induction on n we have $w(F\lceil_{x:=0} \vdash \Box) \le d + w(F) + b$. With Lemma 2.8 on the preceding page we get $w(F \vdash \Box) \le d + w(F) + b$.

This proves the theorem, since for $b' := \lceil \ln s_g(F) / \ln a \rceil$ we have $|R^*| < s_g(F) \le a^{b'}$ and $d + b' = O\left(\sqrt{n \ln s_g(F)}\right)$. $\Box$

Corollary 2.12.

$$s_g(F) = \exp\left(\frac{\Omega\left((F \vdash \Box) - w(F)\right)^2}{n}\right)$$

where $s_g(F)$ is the size of the smallest (general) resolution refutation of F.

The connection between width and size of a proof can be (and is) used to prove lower bounds. In the same work that presented the above connection Ben-Sasson and Widgerson [7] gave new proofs for most lower bounds for general and tree-like resolution known at that time.

Chapter 3

Simulations and Separations

In this chapter we will study the relative complexity of the different resolution refinements and give proofs for the known results. The proofs for linear resolution appear in Chapter 4. The following table and Figure 3.1 summarize the simulations and separations currently known.

	dag	rtrl	reg	ord	tree	lin	sem
neg	$<$	$\not\geq$	$<>$	$<>$	$>$	$\not\geq$	$<$
sem	$<$	$\not\geq$	$<>$	$<>$	$>$	$\not\geq$	
lin	$\leq$	?	$\not\leq$	$\not\leq$	$>$		
tree	$<$	$<$	$<$	$<>$			
ord	$<$	$<$	$<$				
reg	$<$	$\leq$					
rtrl	$\leq$						

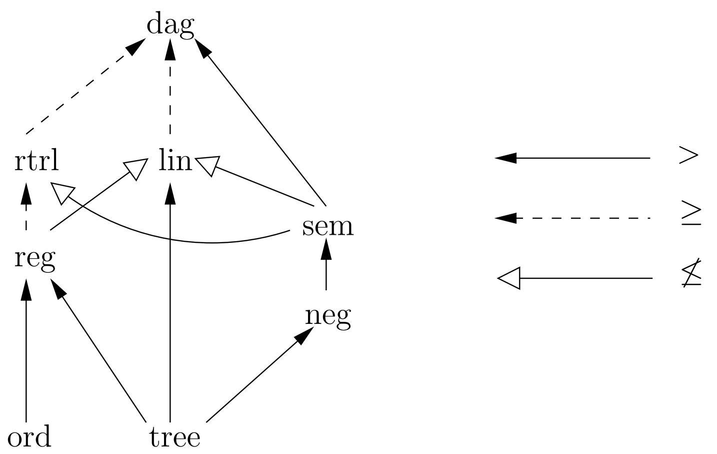

Figure 3.1: Relative Strengths of the Refinements

It is still unknown if linear resolution simulates regular, ordered, semantic or negative resolution. It is unknown if linear resolution simulates regular tree-like resolution with lemmas. It is also unknown if regular tree-like resolution with lemmas simulates linear, semantic, or negative resolution. It is open if there is a separation between general resolution and linear resolution, one between general resolution and regular tree-like resolution with lemmas, or one between regular resolution and regular tree-like resolution with lemmas. At least one of the two latter separations (dag > rtrl or rtrl > reg) must exist, since there is a separation between general and regular resolution.

That rtrl ≤ dag and that lin ≤ dag follows directly from the definitions. That tree < sem, ord < dag, tree < rtrl, and ord < rtrl, follows by transitivity: tree < neg < sem, ord < reg < dag, tree < reg ≤ rtrl, and ord < reg ≤ rtrl, resp.

That semantic (and negative) resolution does not simulate regular tree-like resolution with lemmas follows also from transitivity. That neg ≥ rtrl would imply that neg ≥ reg (because of rtrl ≥ reg), which is a contradiction to the incomparability of regular and negative resolution. That sem ≥ rtrl would imply the same contradiction.

3.1 Tree-like and Ordered Resolution

In this section we prove the incomparability of tree-like and ordered resolution (tree <> ord). This is implied by Corollary 3.7 on page 35 (ord $\not\leq$ tree) and Corollary 3.14 on page 40 (tree $\not\leq$ ord).

3.1.1 Ordered Resolution is Separated from Tree-like Resolution

We first show the separation ord $\not\leq$ tree, following a proof presented by Ben-Sasson et al. [6].

We construct a formula (called *pebbling formula* or short P_G) from a dag $G = (V, E)$ with in-degree 2 as follows. For every $v \in V$, there are two variables $x_0(v)$ and $x_1(v)$. Now P_G consists of the following clauses:

- a *source axiom* $x_0(v) \vee x_1(v)$ for every source v
- two *sink axioms* $\bar{x}_0(v)$ and $\bar{x}_1(v)$ for every sink v
- four *pebbling axioms*

$$x_a(u_1) \wedge x_b(u_2) \to x_0(v) \vee x_1(v)$$

  with $a, b \in \{0, 1\}$ for every non-source node v with the predecessors u_1 and u_2

It is easily seen that this formula is unsatisfiable. P_G has $n := 2|V|$ variables and $m := O(|V|)$ clauses.

Lemma 3.1. *There is an ordered resolution proof R for P_G with $|R| = O(n)$.*

Proof. We first fix some topological ordering $u_1, \ldots, u_n$ of G. Now following this ordering we derive $x_0(u_i) \vee x_1(u_i) =: C$ for every u_i. For a source u_i we already have this in the formula. For any other node we already have derived this clause for each of its predecessors (v_1 and v_2), since they appear earlier in the topological ordering, so we can derive it with a fixed number of steps. Note that we can do this and obey an order compatible with the topological ordering on the u_i.

$$\dfrac{\dfrac{x_0(v_1) \vee x_1(v_1) \qquad \bar{x}_0(v_1) \vee \bar{x}_1(v_2) \vee C}{x_1(v_1) \vee \bar{x}_1(v_2) \vee C}\, x_0(v_1) \qquad \bar{x}_1(v_1) \vee \bar{x}_1(v_2) \vee C}{\bar{x}_1(v_2) \vee C}\, x_1(v_1)$$

$$\dfrac{\dfrac{x_0(v_1) \vee x_1(v_1) \qquad \bar{x}_0(v_1) \vee \bar{x}_0(v_2) \vee C}{x_1(v_1) \vee \bar{x}_0(v_2) \vee C}\, x_0(v_1) \qquad \bar{x}_1(v_1) \vee \bar{x}_0(v_2) \vee C}{\bar{x}_0(v_2) \vee C}\, x_1(v_1)$$

$$\dfrac{\dfrac{x_0(v_2) \vee x_1(v_2) \qquad \bar{x}_1(v_2) \vee C}{x_0(v_2) \vee C}\, x_1(v_2) \qquad \bar{x}_0(v_2) \vee C}{C}\, x_0(v_2)$$

Resolving $x_0(u_n) \vee x_1(u_n)$ with $\bar{x}_0(u_n)$ and $\bar{x}_1(u_n)$ yields the empty clause. This proves the lemma. □

We now prove a lower bound for tree-like resolution refutation on these graphs. We use a game introduced by Pudlák and Impagliazzo [27] to achieve this.

The game is played between two players, *delayer* and *prover*, on a formula F. In each round the prover chooses an unassigned variable. Then the delayer chooses the value 0 or 1 for the variable or $*$. In the latter case the delayer scores one point and the prover chooses the value. The game ends as soon as the assignment falsifies all literals in a clause of F.

Theorem 3.2. *If there is a tree-like refutation of size s of F, then there is a strategy for the prover that prevents the delayer from scoring more than $\lceil \log s \rceil$ points.*

Proof. The prover can keep the following invariant: If the delayer has scored t points, the current assignment falsifies one clause C in the refutation and the part of the refutation rooted in this clause has at most size $s/2^t$. This is obviously true in the beginning.

Now the prover selects the variable v that was eliminated to derive C. If the delayer chooses 1 or 0, one of the clauses C was derived from is falsified, thus the subtree gets smaller (but the delayer does not score any points), and we use this clause as C in the next step. If the delayer chooses $*$, the prover chooses the value of v such that the smaller subtree is rooted in the newly falsified clause. This has at most half of the size of the current tree, thus its size is at most $s/2^{t+1}$ and the invariant still holds.

After the delayer has scored $\lceil \log s \rceil$ points, there is a clause with a subtree of size 1, thus it occurs in F and the game ends. $\square$

Corollary 3.3. *If there is a strategy for the delayer that guarantees him r points, then a tree-like refutation of F must have at least size 2^r.*

Now we use this to prove a lower bound for tree-like refutations of P_G (still following the proof by Ben-Sasson et al. [6]).

Lemma 3.4. *The delayer can score $\Omega(\mathrm{Peb}(G))$ points on a formula P_G.*

The delayer can use the following strategy to achieve this. First he sets $T' := T$ and $S' := S$ where T and S are the sets of the sinks and sources in G.

Now in each round the prover selects a variable $x_i(v)$. The delayer now proceeds as follows.

- If $v \in T'$, he responds 0.
- If $v \in S'$, he responds 1.
- If $v \notin S' \cup T'$ and $\mathrm{Peb}(G, S', T' \cup \{v\}) = \mathrm{Peb}(G, S', T')$, he responds 0 and adds v to T'.
- If $v \notin S' \cup T'$ and $\mathrm{Peb}(G, S', T' \cup \{v\}) < \mathrm{Peb}(G, S', T')$, he responds $*$ and adds v to S'.

We now prove two facts about games where the delayer follows the above strategy.

Lemma 3.5. *After the game $\mathrm{Peb}(G, S', T') \leq 3$.*

Proof. First note that, if $x_i(v)$ is set to 1, v is in S' afterwards, and if both $x_0(v)$ and $x_1(v)$ are set to 0, then v must be in T' afterwards (if the first set one is set via the last case, then v is added to S' and the second one is set to 1).

Since all $s \in S$ are set to 1, no source axiom is violated. Since all $t \in T$ are set to 0, no sink axiom is violated. Thus one of the pebbling axioms must be violated. Suppose it is one of those associated with that node v

which has the predecessors u_1 and u_2. To falsify this, both $x_0(v)$ and $x_1(v)$ must be set to 0, thus $v \in T'$. At least one of $x_i(u_1)$ must be set to 1, thus $u_1 \in S'$ (and analogously $u_2 \in S'$). Now we can pebble from S' to T' by putting a pebble on u_1 and u_2, and then on v, with three pebbles. □

Lemma 3.6. $\mathrm{Peb}(G, S', T') \geq \mathrm{Peb}(G, S, T) - p$ *after each round if the delayer has scored* p *points after this round.*

Proof. Before the first round we have $\mathrm{Peb}(G, S', T') = \mathrm{Peb}(G, S, T)$.

$\mathrm{Peb}(G, S', T')$ only changes in the last case of the above description of the strategy. Since $\mathrm{Peb}(G, S', T' \cup \{v\}) < \mathrm{Peb}(G, S', T')$ at the beginning of the round and by Lemma 1.13 on page 21 we get $\mathrm{Peb}(G, S', T') - 1 \leq \mathrm{Peb}(G, S' \cup \{v\}, T')$. Note that in this case v is added to S' and the delayer scores one point, thus the invariant is preserved. □

Proof (of Lemma 3.4). By the above two lemmas we have

$$3 \geq \mathrm{Peb}(G, S', T') \geq \mathrm{Peb}(G, S, T) - p$$

after the game ends, thus since p is the total number of points scored by the delayer, the delayer scored at least $\mathrm{Peb}(G, S, T) - 3$ points. □

Now we consider a graph G with a high pebbling number (Theorem 1.14 on page 22), i.e., $\mathrm{Peb}(G) \geq \Omega(n/\log n)$. Then from Lemma 3.1 on page 33 and Lemma 3.4 on the facing page together with Corollary 3.3 on the preceding page, the following corollary follows immediately.

Corollary 3.7. *Tree-like resolution is (exponentially) separated from ordered resolution, i.e., ord* $\not\lesssim$ *tree.*

3.1.2 Tree-like Resolution is Separated from Ordered Resolution

Now we show the other direction, i.e., tree $\not\lesssim$ ord, following a proof by Johannsen et al. ([23] and [8]).

Here we use the string of pearls principle, or more exactly a formula $\mathrm{SP}'_{n,m}$ that contradicts this principle. The string of pearls principle says that if from a bag of m pearls that are colored red and blue, n are put on a string and the first one is red and the last one blue, then there must be a red one next to a blue one. The formula $\mathrm{SP}_{n,m}$ has the variables $p_{i,j}$ and r_j with $i \in [n]$ and $j \in [m]$, where $p_{i,j}$ means that pearl j is at position i on the string and r_j means that pearl j is colored red. Now $\mathrm{SP}_{n,m}$ consists of the following clauses.

$$\bigvee_{j=1}^{m} p_{i,j} \qquad i \in [n] \tag{3.1}$$

$$\bar{p}_{i,j} \vee \bar{p}_{i,k} \qquad i \in [n],\ j,k, \in [m], j \neq k \tag{3.2}$$

$$p_{1,j} \to r_j \qquad j \in [m] \tag{3.3}$$

$$p_{n,j} \to \bar{r}_j \qquad j \in [m] \tag{3.4}$$

$$p_{i,j} \wedge r_j \wedge p_{i+1,k} \to r_k \qquad 1 \leq i < n,\ j,k \in [m], j \neq k \tag{3.5}$$

Clauses (3.1) and (3.2) guarantee that a pearl is on each place of the string and that there is only one. Clauses (3.3) force the first pearl to be red, and clauses (3.4) force the last one to be blue. And finally clauses (3.5) say that each red pearl is followed by a red one. Note that we do not require each pearl to occur at most once. This is a difference to the work by Bonet et al. [8], their proof does not work with these clauses. Without these clauses their proof does work. The proof of the upper bound does not need these clauses.

We now prove an upper bound for tree-like refutations of these formulas.

Lemma 3.8. *There are tree-like refutations of* $\mathrm{SP}_{n,m}$ *of size* $m^{O(\log n)}$.

Proof. For $i < h < i' \in [n]$ and $j, j' \in [m]$ we can derive

$$p_{i,j} \wedge r_j \wedge p_{i',j'} \to r_{j'}$$

with tree-like resolution in size $O(m)$ from the above clauses and the $2m$ clauses

$$p_{i,j} \wedge r_j \wedge p_{h,k} \to r_k \text{ and } p_{h,k} \wedge r_k \wedge p_{i',j'} \to r_{j'} \text{ for } k \in [m].$$

We do this by resolving the clauses of each pair with each other eliminating r_k, the resulting m clauses are then resolved with $\bigvee_{l=1}^{m} p_{h,l}$.

We can use this to derive $p_{1,j} \wedge r_j \wedge p_{n,j'} \to r_{j'}$ for $j, j' \in [m]$ in $O(m^{O(\log n)})$ steps. This is done in the following way. To obtain the clause $p_{1,j} \wedge r_j \wedge p_{n,j'} \to r_{j'}$ we build a $2m$-ary tree of clauses, in which each clause $p_{i,j} \wedge r_j \wedge p_{i',j'} \to r_{j'}$ is obtained from $2m$ clauses

$$p_{i,j} \wedge r_j \wedge p_{\left\lceil \frac{i+i'}{2} \right\rceil,k} \to r_k \text{ and } p_{\left\lceil \frac{i+i'}{2} \right\rceil,k} \wedge r_k \wedge p_{i',j'} \to r_{j'} \text{ for } k \in [m]$$

as above. At the leaves the axioms (3.5) are used. This tree has depth $\lceil \log_{2m} n \rceil$, thus it has at most $(2m)^{\lceil \log_{2m} n \rceil + 1}$ nodes. Each node corresponds to a resolution derivation of size $O(m)$. Since there are m^2 of these trees, the derivation of these clauses has size $m^{O(\log n)}$.

These m^2 clauses can be refuted in the following way. We resolve them with the axiom $p_{1,j} \to r_j$ to get the clauses $p_{1,j} \wedge p_{n,j'} \to r_{j'}$ in one step for each of the clauses. These clauses are resolved with $\bigvee_{j=1}^{m} p_{1,j}$, yielding m clauses $p_{n,j'} \to r_{j'}$ in m steps each. Now we resolve these with $p_{n,j'} \to \bar{r}_{j'}$ to get m unit clauses $\bar{p}_{n,j'}$ with m steps. Resolving this with $\bigvee_{l=1}^{m} p_{n,l}$ completes the refutation. This part has size $O(m^2)$. $\square$

We now modify $\mathrm{SP}_{n,m}$ to get formulas $\mathrm{SP}'_{n,m}$, these do have small tree-like refutations as $\mathrm{SP}_{n,m}$, but we can prove a lower bound for ordered resolution.

We call the pearls $j \le n/4$ *special* (we assume $4|n$). For every special pearl and every position on the string, we fix a position in the other half of the string.

$$f(i,j) := \begin{cases} \frac{n}{2} + 2j - 1 & \text{for } 1 \le i \le \frac{n}{2} \\ 2j & \text{for } \frac{n}{2} < i \le n \end{cases}$$

Now for the special pearls (i.e., $j \le n/4$) the clauses (3.3) and (3.4) are replaced by

$$p_{f(1,j),l} \wedge p_{1,j} \to r_j \tag{3.6}$$

$$p_{f(n,j),l} \wedge p_{n,j} \to \bar{r}_j \tag{3.7}$$

for every $l \in [m]$. For $1 \le i < \frac{n}{2}$ the clauses (3.5) are replaced by

$$p_{f(i+1,k),l} \wedge p_{i,j} \wedge r_j \wedge p_{i+1,k} \to r_k \tag{3.8}$$

and for $\frac{n}{2} < i < n$ by

$$p_{f(i,j),l} \wedge p_{i,j} \wedge r_j \wedge p_{i+1,k} \to r_k \tag{3.9}$$

again for every $l \in [m]$ and only the special pearls ($j \le n/4$).

Lemma 3.9. *There are tree-like refutations of* $\mathrm{SP}'_{n,m}$ *with size* $m^{O(\log n)}$.

Proof. By resolving the new clauses with the clauses (3.1), we get a tree-like derivation of any of the removed clauses with size $O(m)$. Thus we need only a tree-like derivation of size $p(m)$ for some polynomial p to derive all of the removed clauses. Thus the upper bound for $\mathrm{SP}_{n,m}$ in Lemma 3.8 on the facing page is an upper bound for $\mathrm{SP}'_{n,m}$, too. $\square$

Now we prove a lower bound for ordered resolution refutation of $\mathrm{SP}'_{n,m}$.

Theorem 3.10. *For sufficiently large n and $m \ge \frac{9}{8}n$, every ordered resolution refutation of* $\mathrm{SP}'_{n,m}$ *has at least size* $2^{\Omega(n \log n)}$.

Proof. For the sake of simplicity we assume $n = 8k$ for some integer k. $N := nm + m$ is the number of variables in the formula. Let an ordering $x_1, \dots, x_N$ of the variables be given, i.e., each of the x_ν is one of the variable $p_{i,j}$ or r_j. Let R be an ordered refutation of $\mathrm{SP}'_{n,m}$ respecting this ordering.

We will now show that R contains at least $k!$ different clauses, which implies the theorem.

We now define $S(i,\nu)$ for a position $i \in [n]$ and $\nu \le N$ to be the set of special pearls ($j \le 2k = n/4$) such that $p_{i,j}$ is among the first ν eliminated variables.

$$S(i,\nu) := \{j \le 2k \mid p_{i,j} \in \{x_1,\dots,x_\nu\}\}$$

Let ν_0 be the smallest index such that $|S(i,\nu_0)| = k$ for some position i, and call this position i_0. Thus $|S(i,\nu_0)| < k$ for $i \neq i_0$. In other words, i_0 is the first position for which k of the $p_{i_0,j}$ with $j \le 2k$ are eliminated.

Now we enumerate the elements of $S(i_0,\nu_0)$ in increasing order denoted by $j_1,\dots,j_k$. For each $1 \le \mu \le k$, let i_μ be the position $f(i_0,j_\mu)$. Note that $i_\mu \neq i_0$.

Further we define $R_\mu := [2k] \setminus S(i_\mu,\nu_0)$, i.e., R_μ is the set of special pearls j with the property that, on every path in the refutation R, the variable $p_{i_\mu,j}$ is eliminated after all the variables p_{i_0,j_κ} for $1 \le \kappa \le k$ have been eliminated. Since $|S(i_\mu,\nu_0)| < k$ we have $|R_\mu| \ge k$.

Definition 3.11. *A* critical assignment *is a total assignment that satisfies all the clauses of* $\mathrm{SP}'_{n,m}$ *except exactly one of the clauses* (3.1). *For a critical assignment α we define:*

- *The unique position $i_\alpha \in [n]$ such that no pearl is placed at position i_α by α, i.e., $\alpha(p_{i_\alpha,j}) = 0$ for every $j \in [m]$.*
- *an injective mapping $m_\alpha : [n] \setminus \{i_\alpha\} \to [m]$ where, for every $i \neq i_\alpha$, $m_\alpha(i)$ is the pearl placed at position i by α, i.e., the unique $j \in [m]$ such that $\alpha(p_{i,j}) = 1$.*

A critical assignment is called 0-critical *if the gap is $i_\alpha = i_0$ and $m_\alpha(i_\mu) \in R_\mu$ for each $1 \le \mu \le k$, and moreover if the following implications hold:*

- *If i_0 is in the left half ($1 \le i \le n/2$), then $j_1,\dots,j_k$ are colored blue (i.e., $\alpha(r_{j_1}) = \ldots = \alpha(r_{j_k}) = 0$).*
- *If i_0 is in the right half ($n/2 < i \le n$), then $j_1,\dots,j_k$ are colored red (i.e., $\alpha(r_{j_1}) = \ldots = \alpha(r_{j_k}) = 1$).*

Note that the positions $i_0,\dots,i_k$ and the pearls $j_1,\dots,j_k$, and thus the notion of 0-critical assignment, only depends on the order of the variables and not on the refutation R.

The lower bound will now be proven in two steps. First we show that there are many 0-critical assignments. Second we will map each 0-critical assignment α to a clause C_α in R, and then show that not too many different assignments can be mapped to one clause, thus there must be many clauses.

The first step is the following lemma.

Lemma 3.12. *For every choice of pairwise distinct pearls $b_1, \ldots, b_k$ with $b_\mu \in R_\mu$ for $1 \leq \mu \leq k$, there is a 0-critical assignment α with $m_\alpha(i_\mu) = b_\mu$ for $1 \leq \mu \leq k$. In particular, there are at least $k!$ 0-critical assignments that disagree on at least one of the values $m_\alpha(i_\mu)$ for $1 \leq \mu \leq k$.*

Proof. First we assign non-special pearls to the positions not yet occupied, i.e., we choose an arbitrary injective mapping from $[n] \setminus \{i_0, i_1, \ldots, i_k\}$ to $[m] \setminus \{b_1, \ldots, b_k, j_1, \ldots, j_k\}$. This is possible, since $m \geq \frac{9}{8}n$.

Now we color the pearls assigned to positions left of the gap red and those assigned to positions right of the gap blue. Formally $\alpha(r_{m_\alpha(i)}) = 1$ for $i < i_0$ and $\alpha(r_{m_\alpha(i)}) = 0$ for $i > i_0$. The special pearls are colored as required by the definition of 0-critical assignment. This is possible, since special pearls can only appear if they are among the pearls $b_1, \ldots, b_k$, and these are in the half that does not contain i_0 (by the definition of i_μ). The remaining pearls can be colored arbitrarily. $\square$

Fix any topological ordering of R. For a 0-critical assignment α, let C_α be the first clause in R such that α does not satisfy C_α and

$$\{j \leq 2k \mid p_{i_0,j} \text{ occurs in } C_\alpha\} = [2k] \setminus \{j_1, \ldots, j_k\}.$$

This clause exists, since there is a path from $\bigvee_{j \in m} p_{i_0,j}$ to the empty clause such that no clause on this path is satisfied by α. The variables $p_{i_0,j}$ with $j \leq 2k$ are eliminated along this path and $p_{i_0,j_1}, \ldots, p_{i_0,j_k}$ are the first of these in the elimination order.

Lemma 3.13. *Let α be a 0-critical assignment. For every $1 \leq \mu \leq k$, the literal $\bar{p}_{i_\mu,l_\mu}$, where $l_\mu := m_\alpha(i_\mu)$, occurs in C_α.*

Proof. We define an assignment α' from α as follows: $\alpha'(p_{i_0,j_\mu}) := 1$ and $\alpha'(x) := \alpha(x)$ for all variables $x \neq p_{i_0,j_\mu}$. By the definition of C_α it does not contain p_{i_0,j_μ}, thus α' does not satisfy C_α. Because of the coloring requirements in the definition of 0-critical assignment there is exactly one clause in $\mathrm{SP}'_{n,m}$ that is not satisfied by α', depending on where gap $i_\alpha = i_0$ is:

$$\begin{aligned}
i_0 = 1\colon &\quad p_{i_\mu,l_\mu} \wedge p_{1,j_\mu} \to r_{j_\mu} \\
1 < i_0 \leq \tfrac{n}{2}\colon &\quad p_{i_\mu,l_\mu} \wedge p_{(i_0-1),h} \wedge p_{i_0,j_\mu} \wedge r_h \to r_{j_\mu} \quad \text{where } h = m_\alpha(i_0 - 1) \\
\tfrac{n}{2} < i_0 < n\colon &\quad p_{i_\mu,l_\mu} \wedge p_{i_0,j_\mu} \wedge p_{(i_0+1),h} \wedge r_{j_\mu} \to r_h \quad \text{where } h = m_\alpha(i_0 + 1) \\
i_0 = n\colon &\quad p_{i_\mu,l_\mu} \wedge p_{n,j_\mu} \to \bar{r}_{j_\mu}
\end{aligned}$$

In all these cases $\bar{p}_{i_\mu,l_\mu}$ occurs in the clause. There is a path from this clause through C_α such that all clauses on the path are falsified by α'. The last variable eliminated on this path is, by the definition of C_α, one of p_{i_0,j_κ} for $1 \leq \kappa \leq k$. Since $l_\mu \in R_\mu$, the variable p_{i_μ,l_μ} appears after p_{i_0,j_κ} by the definition of R_μ. Thus $\bar{p}_{i_\mu,l_\mu}$ was not eliminated on this path, so $\bar{p}_{i_\mu,l_\mu}$ still occurs in C_α. $\square$

We can now finish the proof of the theorem. Take two 0-critical assignments α and β such that $l_\mu := m_\alpha(i_\mu) \neq m_\beta(i_\mu)$ for some $1 \leq \mu \leq k$. Then $\beta(p_{i_\mu,l_\mu}) = 0$. By Lemma 3.13 on the previous page the literal $\bar{p}_{i_\mu,l_\mu}$ occurs in C_α, thus β satisfies C_α and therefore $C_\alpha \neq C_\beta$.

By Lemma 3.12 on the preceding page there are at least $k!$ many 0-critical assignments α that disagree on at least one of the values $m_\alpha(i_\mu)$. Thus R contains at least $k!$ distinct clauses of the form C_α. □

From Lemma 3.9 on page 37 and Theorem 3.10 on page 37 the following corollary follows immediately.

Corollary 3.14. *Ordered resolution is (exponentially) separated from tree-like resolution, i.e., tree $\not\leq$ ord.*

3.2 Negative and Regular Resolution

In this section we will prove the incomparability of negative and regular resolution.

3.2.1 Negative Resolution is Separated from Regular Resolution

To prove the separation neg $\not\leq$ reg we will use a family of formulas based on the ordering principle. This principle states that for a finite set of size n and a total linear ordering on this set, there is a minimal element. First we define the formulas OP_n which are contradictory due to this fact. OP_n contains the variables x_{ij} with $i, j \in [n], i \neq j$, where x_{ij} is assigned to 1 iff $i \prec j$ in the ordering, and the following clauses.

$$x_{ij} \leftrightarrow \bar{x}_{ji} \qquad 1 \leq i < j \leq n \tag{3.10}$$

$$\bar{x}_{i_1 i_2} \vee \bar{x}_{i_2 i_3} \vee \bar{x}_{i_3 i_1} \qquad \text{for any distinct } i_1, i_2, i_3 \in [n] \tag{3.11}$$

$$\bigvee_{k \in [n], k \neq j} x_{kj} \qquad j \in [n] \tag{3.12}$$

Note that the transitivity axioms (3.11) are written differently from the usual form $x_{i_1 i_2} \wedge x_{i_2 i_3} \rightarrow x_{i_1 i_3}$. The symmetric form used here is more convenient for the following proofs and equivalent to the other form because of the clauses (3.10). Note that there are exactly two such transitivity axioms for any set of three distinct $i_1, i_2, i_3 \in [n]$.

Now, following a work by Alekhnovich et al. [2], we modify OP_n to get the formulas $\mathrm{OP}'_{n,\rho}$. These have small positive refutations but (for some ρ) large regular refutations. Let X be the set of the variables x_{ij} used in OP_n.

We define the set $T := \{(i,j,k) \mid i,j,k \in [n], i \neq j \neq k\}$. We fix a mapping ρ from T to X. Now $\mathrm{OP}'_{n,\rho}$ consists of the following clauses.

$$\begin{aligned}
x_{ij} &\leftrightarrow \bar{x}_{ji} && 1 \leq i < j \leq n && (3.13)\\
\bar{x}_{i_1i_2} \vee \bar{x}_{i_2i_3} \vee \bar{x}_{i_3i_1} &\vee \rho(i_1,i_2,i_3) && \text{for } \bar{x}_{i_1i_2} \vee \bar{x}_{i_2i_3} \vee \bar{x}_{i_3i_1} \in \mathrm{OP}_n && (3.14)\\
\bar{x}_{i_1i_2} \vee \bar{x}_{i_2i_3} \vee \bar{x}_{i_3i_1} &\vee \neg\rho(i_1,i_2,i_3) && \text{for } \bar{x}_{i_1i_2} \vee \bar{x}_{i_2i_3} \vee \bar{x}_{i_3i_1} \in \mathrm{OP}_n && (3.15)\\
\bigvee_{k\in[n],k\neq j} & x_{kj} && j \in [n] && (3.16)
\end{aligned}$$

For each transitivity axiom in OP_n there are two axioms in $\mathrm{OP}'_{n,\rho}$, one with the additional literal $\rho(i_1,i_2,i_3)$, the other with $\neg\rho(i_1,i_2,i_3)$. We consider clauses as sets of literals, so we need to assume some arbitrary order of the literals in the original transitivity axioms to get a well-defined order of the arguments of ρ.

We will now prove an upper bound for resolution refutations of $\mathrm{OP}'_{n,\rho}$. This proof is based on a proof by Bonet and Galesi [9] for an upper bound of OP_n.

Theorem 3.15. *There are polynomial size positive resolution refutations of* $\mathrm{OP}'_{n,\rho}$.

Proof. First we introduce some abbreviations.

$$\begin{aligned}
A(i,j,k) &:= \bar{x}_{ij} \vee \bar{x}_{jk} \vee \bar{x}_{ki} \vee \rho(i,j,k) && \text{for any distinct } i,j,k \in [n]\\
A'(i,j,k) &:= \bar{x}_{ij} \vee \bar{x}_{jk} \vee \bar{x}_{ki} \vee \neg\rho(i,j,k) && \text{for any distinct } i,j,k \in [n]\\
B(i,j) &:= x_{ij} \rightarrow \bar{x}_{ji} && i \neq j \in [n]\\
B'(i,j) &:= x_{ij} \leftarrow \bar{x}_{ji} && i \neq j \in [n]\\
C_m(j) &:= \bigvee_{i=1,i\neq j}^{m} x_{ij} && j \in [m], m \in [n]\\
D_k^j(i) &:= C_{k-1}(j) \vee \bar{x}_{ik} && k,i,j \in [n]\\
E_k^j(i) &:= C_{k-1}(j) \vee \bigvee_{l=i}^{k-1} c_{lk} && k,i,j \in [n]
\end{aligned}$$

Note that $A(i,j,k)$, $A'(i,j,k)$, $B(i,j)$, $B'(i,j)$ and $C_n(j)$ are the clauses of $\mathrm{OP}'_{n,\rho}$.

We will now derive (a superset of) $\mathrm{OP}'_{n-1,\rho}$ from $\mathrm{OP}'_{n,\rho}$ for $n > 3$ with $O(n)$ steps. The clauses (3.10) and (3.11) of $\mathrm{OP}'_{n-1,\rho}$ are already present in $\mathrm{OP}'_{n,\rho}$. Thus we need only to derive the clauses (3.12), i.e., $C_{n-1}(j)$ for $j \in [n-1]$.

This is done in the following way. For every $j \in [n-1]$ we first derive:

- $D_n^j(j)$

$$\frac{C_n(j) \quad B(j,n)}{D_n^j(j)}$$

- $D_n^j(i) \vee \rho(n,j,i)$ for every $i \in [n-1] \setminus \{j\}$

$$\frac{\dfrac{C_n(j) \quad A(n,j,i)}{D_n^j(i) \vee \bar{x}_{ji} \vee \rho(n,j,i)} \qquad B'(j,i)}{D_n^j(i) \vee \rho(n,j,i)}$$

- $D_n^j(i) \vee \neg\rho(n,j,i)$ for every $i \in [n-1] \setminus \{j\}$

$$\frac{\dfrac{C_n(j) \quad A'(n,j,i)}{D_n^j(i) \vee \bar{x}_{ji} \vee \neg\rho(n,j,i)} \qquad B'(j,i)}{D_n^j(i) \vee \neg\rho(n,j,i)}$$

From these clauses we now derive for every $j \in [n-1]$ the clause $C_{n-1}(j)$. First we derive $E_n^j(2)$.

$$\frac{\dfrac{C_n(n) \quad D_n^j(1) \vee \rho(n,j,i)}{E_n^j(2) \vee \rho(n,j,i)} \qquad \dfrac{C_n(n) \quad D_n^j(1) \vee \neg\rho(n,j,i)}{E_n^j(2) \vee \neg\rho(n,j,i)}}{E_n^j(2)}$$

Now we derive $E_n^j(i)$ using $E_n^j(i-1)$ for $i = 3, \ldots, n-1$ and $i \neq j+1$.

$$\frac{\dfrac{E_n^j(i-1) \quad D_n^j(i-1) \vee \rho(n,j,i-1)}{E_n^j(i) \vee \rho(n,j,i-1)} \qquad \dfrac{E_n^j(i-1) \quad D_n^j(i-1) \vee \neg\rho(n,j,i-1)}{E_n^j(i) \vee \neg\rho(n,j,i-1)}}{E_n^j(i)}$$

For $i = j+1$ we derive $E_n^j(i)$ in the following way.

$$\frac{E_n^j(i-1) \quad D_n^j(i-1)}{E_n^j(i)}$$

Finally we can derive $C_{n-1}(j)$ using $E_n^j(n-1)$.

$$\frac{\dfrac{E_n^j(n-1) \quad D_n^j(n-1) \vee \rho(n,j,n-1)}{C_{n-1}(j) \vee \rho(n,j,n-1)} \qquad \dfrac{E_n^j(n-1) \quad D_n^j(n-1) \vee \neg\rho(n,j,n-1)}{C_{n-1}(j) \vee \neg\rho(n,j,n-1)}}{C_{n-1}(j)}$$

This derivation is positive since $C_n(j)$, $B'(j,i)$, $E_n^j(i)$ are positive clauses, and one of these is used in every step. Thus we can positively derive $\mathrm{OP}'_{3,\rho}$ from $\mathrm{OP}'_{n,\rho}$ in $O(n^2)$ steps. Since there is, due to the completeness of positive resolution, a positive refutation of $\mathrm{OP}'_{3,\rho}$, we can refute $\mathrm{OP}'_{n,\rho}$ in $O(n^2)$ steps. $\square$

Before we prove the lower bound for regular refutations, we first introduce some definitions and lemmas. This follows (with some small clarifications) the work by Alekhnovich et al. [2]. The proof is also due to them.

Definition 3.16. *For an assignment α let $Supp(\alpha)$ be the set of all $i \in [n]$ such that α assigns x_{ij} or x_{ji} to a value for some j.*

Definition 3.17. *An assignment is called* critical *if it assigns all variables x_{ij} to a value and falsifies only one of the clauses (3.12) of OP_n.*

Let $S \subset [n]$. We call an assignment α a partial critical assignment for S *iff it fulfills the following two conditions:*

- *$\{x_{ij} \mid i, j \in S\}$ is the domain of α.*
- *$C\lceil_\alpha \neq 0$ holds for every clause $C \in \mathrm{OP}_n$.*

A critical assignment specifies a linear ordering of $[n]$ with one minimal element. A partial critical assignment for S corresponds to an ordering of S. There is a bijective mapping between the orderings of S and the partial critical assignments for S. Note that for a partial critical assignment α for S, $Supp(\alpha) = S$.

Lemma 3.18. *Let α be a partial critical assignment with $|Supp(\alpha)| < n-2$ and x_{ij} unassigned by α. Then for $\varepsilon \in \{0,1\}$, α can be extended to a (partial) critical assignment α' with $|Supp(\alpha')| \leq |Supp(\alpha)|+2$ such that $\alpha'(x_{ij}) = \varepsilon$.*

Proof. α induces a linear ordering of $Supp(\alpha)$. We need to add one or two new elements to $Supp(\alpha)$ (depending on whether both i and j or only one of them is in $Supp(\alpha)$). We can clearly choose $i \prec j$ or $j \prec i$ and insert the element(s) into the ordering correctly. We choose depending on ε. For this ordering we have again a corresponding assignment α'. The inequality follows from the fact that α' talks about at most two elements more than α. $\square$

Lemma 3.19. *Let α be a partial critical assignment with $|Supp(\alpha)| \leq n/100$ and i_1, i_2, i_3 distinct elements of $[n] \setminus Supp(\alpha)$. Then α can be extended to a total assignment that satisfies all clauses of OP_n except*

$$\bar{x}_{i_1 i_2} \vee \bar{x}_{i_2 i_3} \vee \bar{x}_{i_3 i_1}.$$

Proof. We extend the ordering induced by α on $Supp(\alpha)$ to a total ordering on $[n]$. First we extend α to an arbitrary partial critical assignment on $[n] \setminus \{i_1, i_2, i_3\}$. Then we set $i_1 \prec i_2 \prec i_3 \prec i_1$ and let i_1, i_2, i_3 be greater than all other elements. $\square$

Lemma 3.20. *For sufficiently large n, there is a ρ such that for every S and x_{ij}, where $S \subset [n]$ and $|S| \leq n/100$, and x_{ij} one of the variables of OP_n, there are three distinct elements $i_1, i_2, i_3 \in [n] \setminus S$ such that $\rho(i_1, i_2, i_3) = x_{ij}$.*

Proof. Fix a set S with $|S| \leq \varepsilon n$, $\varepsilon = 1/100$, and some x_{ij}. The probability that there are no distinct $i_1, i_2, i_3 \in [n] \setminus S$ such that $\rho(i_1, i_2, i_3) = x_{ij}$ is at most

$$
\begin{aligned}
& \left(\frac{n(n-1)-1}{n(n-1)}\right)^{\binom{n-\varepsilon n}{3}} \\
\leq \quad & \left(1 - \frac{1}{n^2}\right)^{\binom{n-\varepsilon n}{3}} \\
\leq \quad & \left(1 - \frac{1}{n^2}\right)^{(n-\varepsilon n)^3/12} \\
= \quad & \left(1 - \frac{1}{n^2}\right)^{n^3(1-\varepsilon)^3/12} \\
\leq \quad & e^{-n(1-\varepsilon)^3/12} \qquad \text{(remember } \left(1-\frac{1}{x}\right)^x \leq e^{-1}\text{)}
\end{aligned}
$$

The second inequality holds since $(n - \varepsilon n - 2) > \frac{n-\varepsilon n}{\sqrt{2}}$ (for sufficiently large n) and thus

$$
\begin{aligned}
& \binom{n-\varepsilon n}{3} \\
= \quad & \frac{(n-\varepsilon n)(n-\varepsilon n-1)(n-\varepsilon n-2)}{6} \\
\geq \quad & \frac{(n-\varepsilon n)\frac{(n-\varepsilon n)}{\sqrt{2}}\frac{(n-\varepsilon n)}{\sqrt{2}}}{6} \\
= \quad & (n-\varepsilon n)^3/12.
\end{aligned}
$$

Thus the probability that for some S with $|S| \leq \varepsilon n$ and some x_{ij} there are no distinct $i_1, i_2, i_3 \in [n] \setminus S$ such that $\rho(i_1, i_2, i_3) = x_{ij}$ is at most

$$
\begin{aligned}
& n^2 \cdot \binom{n}{\varepsilon n} \cdot e^{-n(1-\varepsilon)^3/12} \\
= \quad & n^2 \cdot \frac{n!}{(n-\varepsilon n)!(\varepsilon n)!} \cdot e^{-n(1-\varepsilon)^3/12} \\
\leq \quad & n^3 \cdot (1/(1-\varepsilon))^{n-\varepsilon n} \cdot (1/\varepsilon)^{\varepsilon n} \cdot e^{-n(1-\varepsilon)^3/12} \\
= \quad & e^{3\ln n} \cdot e^{(n-\varepsilon n)\ln(1/(1-\varepsilon))} \cdot e^{\varepsilon n \ln(1/\varepsilon)} \cdot e^{-n(1-\varepsilon)^3/12}
\end{aligned}
$$

The third line follows from Stirling's approximation ($n! \geq (n/e)^n$ in the denominator and $n! \leq n(n/e)^n$ in the numerator). This probability is strictly smaller than 1 if

$$3\ln n + (n-\varepsilon n)\ln(1/(1-\varepsilon)) + \varepsilon n \ln(1/\varepsilon) - n(1-\varepsilon)^3/12 < 0$$

holds. This is the case for $\varepsilon = 1/100$ if n is sufficiently large. Thus the probability that a randomly chosen ρ satisfies the lemma is strictly larger than 0. And thus such a ρ exists. $\square$

Theorem 3.21. *For a sufficiently large n, there is a ρ such that any regular refutation of $\mathrm{OP}'_{n,\rho}$ has a size greater than $2^{n/200}$.*

Proof. First we fix some ρ that does satisfy the condition in Lemma 3.20 on page 43. Now we show that a regular refutation R of $\mathrm{OP}'_{n,\rho}$ contains a set of paths, each of which contains a clause that is contained in none of the others, and there are at least $2^{n/200}$ such paths.

We will define for each node ν on such a path a partial critical assignment α_ν that falsifies the clause ν. We build this set incrementally starting with one path $(\Box)$ and $\alpha_\Box = \emptyset$. All the paths will start in $\Box$.

Now assume there are already l paths defined, that end with $\nu_1, \ldots, \nu_l$. For every path where $|Supp(\alpha_{\nu_k})| < n/100$, we do the following:

- ν_k is an axiom: This cannot happen, since ν_k is falsified by a partial critical assignment and these assignments falsify none of the axioms.
- ν_k was derived by copying the label from its only predecessor: We extend the path with this node and do not change the assignment.
- ν_k was derived by the elimination of x_{ij}, and α_{ν_k} assigns x_{ij} to a value: We extend our path with the predecessor that is not satisfied by α_{ν_k} and do not change the assignment.
- Otherwise we call ν_k a *branching node*, and it is derived by the elimination of x_{ij} from ν_{k_0} and ν_{k_1}, and α_{ν_k} does not assign x_{ij} to a value. In this case we extend the path in two ways by appending ν_{k_0} or ν_{k_1} (here we increase the number of paths). By Lemma 3.18 on page 43 we can extend α_{ν_k} to $\alpha_{\nu_{k_0}}$ such that $\alpha_{\nu_{k_0}}$ is a partial critical assignment that falsifies ν_{k_0} with $|Supp(\alpha_{\nu_{k_0}})| \leq |Supp(\alpha_{\nu_k}) + 2$. In the same way we get the assignment $\alpha_{\nu_{k_1}}$.

We repeat this until every path ends in a node ν with $|Supp(\alpha_\nu)| \geq n/100$. Since the value of $|Supp(\alpha_\nu)|$ is increased at most by 2 in any branching node, every path must have at least $n/200$ branching nodes. Hence there are $2^{n/200}$ distinct paths. To prove the theorem we only need to show that any two paths do not have any nodes in common after they diverged.

We prove this by contradiction. Assume two paths diverge in node ν_1 and merge again in ν_2. Let x_{ij} be the variable that is eliminated to derive ν_1. The assignments on both paths assign x_{ij} to different values, and extensions of both falsify ν_2. Hence ν_2 cannot contain x_{ij} or $\bar{x}_{ij}$. Furthermore, no clause that ν_2 is derived from can contain the variable x_{ij} since it is eliminated between ν_2 and the empty clause and R is regular.

Now we choose $i_1, i_2, i_3 \in [n] \setminus Supp(\alpha_{\nu_2})$ by Lemma 3.20 on page 43 such that $\rho(i_1, i_2, i_3) = x_{ij}$. Now we extend α_{ν_2} by Lemma 3.19 on page 43 to a total assignment α' in such a way that all axioms of $\mathrm{OP}'_{n,\rho}$ except $\bar{x}_{i_1 i_2} \vee \bar{x}_{i_2 i_3} \vee \bar{x}_{i_3 i_1} \vee x_{ij}$ or $\bar{x}_{i_1 i_2} \vee \bar{x}_{i_2 i_3} \vee \bar{x}_{i_3 i_1} \vee \bar{x}_{ij}$ are satisfied. α' falsifies

ν_2 since α' is an extension of α_{ν_2}. Hence ν_2 must be derived from some axiom that is falsified by α'. But all violated axioms contain x_{ij}, which is a contradiction, since ν_2 cannot contain x_{ij}. □

Remark 3.22. *Note that a simplified version of this proof can be used to prove a lower bound of $2^{\frac{n}{2}-1}$ for tree-like refutations of* OP_n. *According to Johannsen [25], there are short regular proofs for* OP_n, *so these formulas can be used to prove the separation between regular and tree-like resolution.*

Now we define the formulas $\overline{\mathrm{OP}}'_{n,\rho}$ from the formulas $\mathrm{OP}'_{n,\rho}$. $\overline{\mathrm{OP}}'_{n,\rho}$ is identical to $\mathrm{OP}'_{n,\rho}$ but with the sign of every literal flipped, i.e., every literal x^ε is replaced by $x^{1-\varepsilon}$. $\overline{\mathrm{OP}}'_{n,\rho}$ is clearly unsatisfiable. The lower bound (Theorem 3.21 on the previous page) for regular refutations of $\mathrm{OP}'_{n,\rho}$ applies to $\overline{\mathrm{OP}}'_{n,\rho}$ as well, since any regular refutation of $\overline{\mathrm{OP}}_{n,\rho}$ can be transformed to a regular refutation of $\mathrm{OP}'_{n,\rho}$ by flipping the signs of all literals. A small positive refutation of $\mathrm{OP}'_{n,\rho}$ (Theorem 3.15 on page 41) can, in the same way, be transformed into a negative refutation of $\overline{\mathrm{OP}}'_{n,\rho}$. This implies the following corollary.

Corollary 3.23. *Regular resolution does not simulate negative resolution, i.e.,*

$$\textit{neg} \not\leq \textit{reg}.$$

3.2.2 Negative Resolution does Not Simulate Regular Resolution

The separation neg $\not\geq$ reg is implied by reg $\not\leq$ sem, as proven in Section 3.11 and neg $\leq$ sem.

3.3 Regular and General Resolution

In this section we will show reg $<$ dag. That reg $\leq$ dag follows immediately from the definition.

The separation reg $\not\geq$ dag was first proven by Goerdt [19]. It was later improved by Alekhnovich et al. [2]. It follows directly from Corollary 3.23 and dag $\geq$ neg.

3.4 Tree-like and General Resolution

Here we show tree $<$ dag. That tree $\leq$ dag follows immediately from the definition, and the separation tree $\not\geq$ dag follows via transitivity from Corollary 3.7 on page 35 (ord $\not\leq$ tree) and ord $\leq$ dag.

3.5 Tree-like and Regular Resolution

To prove tree $<$ reg, we first need tree $\leq$ reg. This follows from Theorem 1.9 on page 16, which shows that every tree-like resolution refutation can be converted to a regular tree-like refutation without increasing its size.

In Section 3.1.1 we proved tree $\not\geq$ ord. Since every ordered proof is also regular, this implies tree $\not\geq$ reg. For another way to prove this separation see Remark 3.22 on the facing page.

3.6 Ordered and Regular Resolution

Here we show ord $<$ reg. The simulation ord $\leq$ reg follows immediately from the definition.

The separation ord $\not\geq$ reg was first proven by Goerdt [17]. A better separation results from the separation tree $\not\leq$ ord, as proven in Section 3.1.2. Together with the fact that regular resolution simulates tree-like resolution (Theorem 1.9 on page 16), this exponentially separates regular from ordered resolution.

3.7 Regular Resolution and Regular Tree-like Resolution with Lemmas

That reg $\leq$ rtrl follows directly from the fact that for every regular refutation there is a tree so that said refutation is a regular tree-like refutation with lemmas.

Theorem 3.24. *Regular tree-like refutation with lemmas simulates regular resolution.*

Proof. Let R be a regular[1] proof of the formula F. We will construct a rtrl proof R' with $|R'| \leq 3 \cdot |R|$.

We do a depth first search starting with the empty clause and visiting the left predecessor first. After the DFS is finished, we split all non-tree edges in two by adding a new node in the middle. The new node (a lemma) is labeled by the clause on the starting node of the original edge. The edge ending in the original ending node is marked as part of the tree. The result is a regular tree-like refutation with lemmas.

The marked edges form a tree, since we just added edges pointing to new nodes to the tree resulting from the DFS.

All edges to lemmas are in fact from left to right, since the edge we split was not part of the DFS tree, thus the starting node was already in the tree and, since we visited the left part first, left of the current node.

[1] Note that the same proof shows that tree-like resolution with lemmas simulates general resolution.

And since R is regular, the tree is regular, too, since we did not add any new eliminations. Since every node has at most two predecessors and we only duplicate these, R' can have at most size $3 \cdot |R|$. □

There is currently work in progress to prove a separation between regular resolution and regular tree-like resolution with lemmas [22].

3.8 Tree-like and Negative Resolution

In this section we will prove tree < neg. First we prove that tree-like resolution is simulated by negative resolution (tree ≤ neg). This fact was already known [11], but a proof could not be found in the available literature.

Theorem 3.25. *If there is a tree-like refutation R of F, then there is a negative refutation of F with a size of at most $l \cdot |R|$, where $l := \min(|R|, |F|)$.*

Proof. Let $F_x \subset F$ be the set of all clauses containing x, and let $F_{\bar{x}} \subset F$ be the set of all clauses containing $\bar{x}$. Let s be the size of R. We prove the theorem by induction on s. The case $s = 1$ is obvious.

Otherwise R ends with:

$$\frac{\bar{x} \quad x}{\Box}$$

Then there are tree-like derivations R_x and $R_{\bar{x}}$ of x and $\bar{x}$ from F with $|R_x| + |R_{\bar{x}}| + 1 = s$. By Theorem 1.7 on page 16 there are refutations R'_x and $R'_{\bar{x}}$ of $F\lceil_{x=0}$ and $F\lceil_{x=1}$ with $|R'_x| \le |R_x|$ and $|R'_{\bar{x}}| \le |R_{\bar{x}}|$.

By the induction hypothesis there is a negative refutation $P_{\bar{x}}$ of $F\lceil_{x=1}$ with $|P_{\bar{x}}| \le l|R'_{\bar{x}}|$. If no clause of $F_{\bar{x}}\lceil_{x=1}$ is used in $P_{\bar{x}}$, then $P_{\bar{x}}$ is a refutation of F, and we are done. Otherwise we add $\bar{x}$ to all clauses of $F_{\bar{x}}\lceil_{x=1}$ that are used in $P_{\bar{x}}$, and all clauses derived from these (directly or indirectly), to obtain a negative derivation of $\bar{x}$ from F of size at most $l|R'_{\bar{x}}|$.

By the induction hypothesis there is a negative refutation P_x of $F\lceil_{x=0}$ with $|P_x| \le l|R'_x|$. Each of these clauses C in $F_x\lceil_{x=0}$ that are used by P_x can be derived in one step from the clause $C \vee x$ in F using $\bar{x}$ as derived before. We need at most l steps for this. By this we get a negative refutation of F with a size of at most

$$l|R'_{\bar{x}}| + l|R'_x| + l \le l|R_{\bar{x}}| + l|R_x| + l = l(|R_{\bar{x}}| + |R_x| + 1) = ls.$$

□

The separation tree $\not\geq$ neg follows directly from Corollary 3.23 on page 46 and tree ≤ reg (Section 3.5 on the preceding page). It was first proven by Bonet and Galesi [9].

3.9 Negative and Semantic Resolution

In this section we show neg $<$ sem. It follows immediately from the definition that neg $\leq$ sem.

To show the separation we will reuse the pebbling formulas P_G defined on page 32 in Section 3.1.1.

Lemma 3.26. *There is a positive (and thus semantic) resolution refutation R of P_G with $|R| = O(n)$.*

Proof. We derive $x_0(v) \vee x_1(v) =: C$ for every v, following some topological ordering. For a source v we already have this in the formula. For any other node we already have derived this clause for each of its predecessors (v_1 and v_2), so we can derive it with a fixed number of steps.

$$\cfrac{\cfrac{x_0(v_1) \vee x_1(v_1) \quad \bar{x}_0(v_1) \vee \bar{x}_0(v_2) \vee C}{x_1(v_1) \vee \bar{x}_0(v_2) \vee C} \qquad x_0(v_2) \vee x_1(v_2)}{x_1(v_1) \vee x_1(v_2) \vee C}$$

$$\cfrac{\cfrac{x_1(v_1) \vee x_1(v_2) \vee C \quad \bar{x}_1(v_1) \vee \bar{x}_0(v_2) \vee C}{\bar{x}_0(v_2) \vee x_1(v_2) \vee C} \qquad x_0(v_2) \vee x_1(v_2)}{x_1(v_2) \vee C}$$

In the same way we derive $x_0(v_1) \vee C$.

$$\cfrac{\cfrac{x_1(v_2) \vee C \quad \bar{x}_0(v_1) \vee \bar{x}_1(v_2) \vee C}{\bar{x}_0(v_1) \vee C} \qquad x_0(v_1) \vee C}{C}$$

Resolving $x_0(v_n) \vee x_1(v_n)$ with $\bar{x}_0(v_n)$ and $\bar{x}_1(v_n)$ yields the empty clause. This proves the lemma. $\square$

The lower bound for negative resolution on this formulas was shown by Buresh-Oppenheim et al. [10]. First we define simplified pebbling formulas P'_G.

P'_G is constructed from a dag $G = (V, E)$ as follows. For every $v \in V$, there is one variable $x(v)$. Now P'_G consists of the following clauses:

- a *source axiom* $x(v)$ for every source v
- a *sink axiom* $\bar{x}(v)$ for every sink v
- a *pebbling axiom*

$$x(u_1) \wedge x(u_2) \rightarrow x(v)$$

for every non-source node v with the predecessors u_1 and u_2

Lemma 3.27. *Let R be a negative resolution refutation of P_G of size s. Then there is a negative resolution refutation R' of P'_G such that every clause in R' contains at most $\log_2 s$ negative literals.*

Proof. We obtain the refutation R' from R by applying a restriction ρ with the following property: For every node v, one of the variables $x_0(v)$ or $x_1(v)$ is set to 0, the other one is not assigned to a value. It is clear that R' is, after renaming the unset variables $x_\varepsilon(v)$ to $x(v)$, a refutation of P'_G.

Now we consider a randomly taken ρ where the choice for each node is independent and where each of the two variables is chosen with equal probability. If a clause in R contains k negative literals, then it is left unsatisfied by ρ with a probability of at most 2^{-k}. Thus, the probability that R' still contains some clause with more than $\log_2 s$ negative literals is, since there at most s such clauses in R, at most $s \cdot 2^{-(\log_2(s)+1)}$, which is less than one. Therefore there is a ρ such that R' contains only clauses with at most $\log_2 s$ negative literals. $\square$

Lemma 3.28. *If P'_G has a negative refutation such that every clause contains at most k negative literals, then* $\mathrm{Peb}(G) \leq k + 1$.

Proof. Note that P'_G contains only clauses with at most one positive literal. Thus resolving a negative clause with an axiom yields a negative clause. The only negative clauses in P'_G are the sink clauses, namely $\bar{x}(t)$ with t a sink in G. Every negative refutation forms a list $\bar{x}(t) = C_0, C_1, \ldots, C_{l-1}, C_l = \Box$, where each clause C_i with $i > 0$ is derived from C_{i-1} and an axiom. Note that, while a clause could potentially be used with multiple axioms, only one of the results can be used on a path to $\Box$.

From this list we construct a pebbling of G in the following way: We define a sequence $(D_i)_{0\leq i\leq l}$ of sets of vertices with $D_i := \{v \mid \bar{x}(v) \in C_{l-i}\}$. This sequence induces a pebbling of G. It is obvious that $D_0 = \emptyset$ and $D_l = \{t\}$.

If C_i is obtained by resolving on $x(v)$ with a source axiom, then $D_{l-i+1} = D_{l-i} \cup \{v\}$, i.e., a pebble is put on a source. If C_i is obtained by resolving on $x(v)$ with a pebbling axiom $x(v_1) \wedge x(v_2) \to x(v)$, then D_{l-i} has pebbles on v_1 and v_2. Hence we can obtain D_{l-i+1} by putting a pebble on v and removing the pebbles from v_1 and v_2 afterwards. The intermediate set contains $|D_{l-i}| + 1 = |C_i| + 1$ nodes. This pebbling has thus a complexity of $k + 1$. $\square$

These two lemmas immediately imply the following corollary.

Corollary 3.29. *Any negative refutation R of P_G has a size of at least* $2^{\Omega(\mathrm{Peb}(G))}$.

Since there are graphs with pebbling numbers of at least $\Omega(n/\log n)$ (Theorem 1.14 on page 22), this corollary together with the upper bound in Lemma 3.26 on the preceding page yields the separation and thus neg < sem.

3.10 Ordered and Semantic Resolution

The incomparability of ordered and semantic resolution (ord <> sem) is proven in this section.

3.10.1 Ordered Resolution is Separated from Semantic Resolution

This separation was proven by Buresh-Oppenheim and Pitassi [11]. The formulas are based on the simplified pebbling formulas introduced in Section 3.9 on page 49. Note that we use here dags with arbitrary in-degree, as long as it is bounded by $O(\log |V|)$. First we introduce the formulas, then we introduce some additional definitions and lemmas.

PP_G is constructed from a dag $G = (V, E)$ as follows. For every $v \in V$, there is one variable v. We will identify each node with its variable. Now PP_G consists of the following clauses:

- a *source axiom* v for every source v
- a *sink axiom* $\bar{v}$ for every sink v
- a *pebbling axiom*
$$\bigwedge_{i=1}^{k} u_i \to v$$
for every non-source node v with the predecessors u_i, $i = 1, \dots, k$

Definition 3.30. *Let $G = (V, E)$ be a dag and $S, T \subset V$. We define a* c-pebbling *from S to T in the same way as a pebbling*[2] *from S to T, but for the last pebble-configuration C_k we replace the condition $C_k \cap T \neq \emptyset$ with $C_k \supseteq T$, i.e., a c-pebbling needs to put pebbles on the complete set T, not only on one node in T. We will call a c-pebbling from the sources of G to T a c-pebbling of T. And we will write c-pebbling of t instead of c-pebbling of $\{t\}$.*

$\mathrm{cPeb}(G, S, T)$ *and* $\mathrm{cPeb}(G)$ *are defined as* $\mathrm{Peb}(G, S, T)$ *and* $\mathrm{Peb}(G)$*, resp., but using a c-pebbling instead of a pebbling.*

Observation 3.31. *Note that $T \subset T'$ implies* $\mathrm{cPeb}(G, S, T) \leq \mathrm{cPeb}(G, S, T')$*. This does not hold for normal pebbling.*

Definition 3.32. *We call a dag G* layered *if the nodes can be assigned to* layers *1 to m, while obeying the following rules. Each node is assigned to exactly one layer, all sinks are in layer 1, all sources are in layer m, and all edges are from nodes in layer i to nodes in layer $i - 1$. We will write $V^{(i)}$ to denote the nodes in layer i.*

[2] See Section 1.2.5 on page 21 for the definition of pebbling.

We call a layered dag G with $\sum_{i=1}^{m} i$ nodes pyramid-like *if there are exactly i nodes on layer i.*

*We call a layered dag r-*expanding,*, if for any layer $i < m$ and any subset S of $V^{(i)}$ with $|S| \leq \lceil |V^{(i)}|/r \rceil$, there are more than $|S|$ nodes in layer $i+1$ with edges to nodes in S.*

We will use $Pyr(m,d)$ to denote the following distribution on pyramid-like dags with m layers. For each node v on layer $1 \leq i < m$ there are d nodes on layer $i+1$ chosen randomly and independently with replacement. The set of the chosen nodes is then the set of v's parents.

Lemma 3.33. *Let G be an r-expanding layered graph with m layers, such that each layer i contains at least $r \cdot i$ nodes. Then for any node v in layer 1 we have* $\mathrm{cPeb}(G, \emptyset, \{v\}) \geq m$.

Proof. Let t be a node in layer 1. In any c-pebbling of t there must be a configuration that puts a pebble on at least one node in each path from a source to t. The second to last configuration is such a configuration, since there must be a pebble on each predecessor of t. Let C_i be the first such configuration. In C_{i-1} there is is a path p from a source s to to t that does not contain a pebbled node. Since every node in p has at least one unpebbled predecessor, the step from C_{i-1} to C_i is to put a pebble on s. Let $p = (s = p_m, \ldots, p_1 = t)$. Now each path from a source other than s to a node p_i must contain at least one pebbled node, since each such path together with the path from p_i to t is a path from a source to t which is pebbled in C_i, and as we have shown, the only pebble on p pebbles s. Now we show that there are at least $m-1$ such paths which are vertex-disjoint. Together with the pebble on s we get the lower bound on the $\mathrm{cPeb}(G, \emptyset, \{v\})$ we want.

We construct a set of these paths as follows. First let $X_1 := \{p_1\}$. Since G is r-expanding, there is at least one predecessor of p_1 other than p_2. Let $pred(p_1)$ be one. Now let $X_2 := \{pred(p_1), p_2\}$. We want each X_i, $i < m$ to be a subset of $V^{(i)}$ with size i. We construct the further X_is as follows: Since X_i has at most $1/r$ times the size of $V^{(i)}$ and G is r-expanding, Hall's theorem guarantees that there is a matching from X_i into the set of its predecessors minus p_{i+1}. Let X_{i+1} be the set of this matched predecessors plus p_{i+1}. Finally the matchings used at each step form vertex-disjoint paths from each of the $m-1$ nodes in X_{m-1} to the nodes in p. □

Definition 3.34. *For every assignment α we define a function f_α. f_α flips the signs of all literals whose variables are set to one by α, i.e., $f_\alpha(F)$ results from F by replacing x by $\bar{x}$ and vice versa for all x with $\alpha(x) = 1$.*

Note that if R is an ordered resolution refutation of F, then $f_\alpha(R)$ is an ordered refutation of $f_\alpha(F)$.

Lemma 3.35. *For every formula F and every assignment β, for a semantic refutation R of F, there is a semantic refutation of the same width and size of $f_\beta(F)$.*

Proof. Let α be the assignment used by R, i.e., α is the assignment given with R that satisfies the constraint for semantic resolution. $f_\beta(R)$ is now a semantic refutation of $f_\beta(F)$ using the assignment $\alpha \oplus \beta$. Clearly $f_\beta(R)$ has the same size and width as R. □

Definition 3.36. *Let F be a formula using the variables $x_1, \ldots, x_n$, and let $x'_1, \ldots, x'_n$ be a disjoint set of variables. Now we define the* xorification *of a literal x_i^ε as the formula $\mathrm{XOR}(x_i^\varepsilon) = x_i \oplus x'_i \oplus \varepsilon$. The xorification $\mathrm{XOR}(C)$ of a clause $C = \bigvee_{j=1}^k l_j$ is the formula in CNF that is equivalent to $\bigvee_{j=1}^k \mathrm{XOR}(l_j)$. The xorification of F is the conjunction of the xorifications of its clauses.*

Note that $\mathrm{XOR}(F)$ of an unsatisfiable formula F with width k and m clauses is an unsatisfiable formula with width $2k$ and at most $2^k m$ clauses, since $\mathrm{XOR}(C)$ consists of at most 2^k clauses, because of the translation to CNF. Furthermore, for an assignment α that assigns for every i exactly one variable, either x_i or x'_i, to a value, $\mathrm{XOR}(F)\lceil_\alpha$ is, after renaming variables, equivalent to $f_\beta(F)$ for some assignment β.

There is a connection between the size and the width of semantic refutation of the formulas resulting from xorification:

Lemma 3.37. *Let F be a formula, R_1 a minimum-width semantic refutation of F, and R_2 a minimum-size semantic refutation of $\mathrm{XOR}(F)$. Then R_2 has at least size $\exp(\Omega(w))$, where w is the width of R_1.*

Proof. Let s be the size of R_2 and $x_1, \ldots, x_n$ the variables occurring in F. Now we choose an assignment α randomly. For each i we choose uniformly one of the variables x_i and x'_i as well as a truth value for the chosen variable. A clause C from R_2 with width at least w appears in $R_2\lceil_\alpha$ with probability at most $\left(\frac{3}{4}\right)^w$. Hence the expected number of such wide clauses that remain in $R_2\lceil_\alpha$ is at most $s \cdot \left(\frac{3}{4}\right)^w$.

Now we prove by contradiction that s must be at least $\left(\frac{4}{3}\right)^w$. Assume $s < \left(\frac{4}{3}\right)^w$. Then this expectation value is smaller than 1, thus there exists a ρ such that $R_2\lceil_\alpha$ does not contain any wide clause. Since $R_2\lceil_\alpha$ contains a semantic refutation of $\mathrm{XOR}(F)\lceil_\alpha$ (see the proof of Theorem 1.7 on page 16), there is by Lemma 3.35 also a refutation of F with width less than w. This contradicts the minimal width of R_1. □

Definition 3.38. *Let $F_1, \ldots, F_l$ be formulas using the variables $x_1, \ldots, x_n$ with $l = 2^c$. Let $Y = \{y_0, \ldots, y_{c-1}\}$ be a disjoint set of variables. For*

$0 \leq i < l$, let $b(i)$ be the interpretation of i as a bit-string of length c and let $b(i)(j)$ be the j-th bit in this string. Now we define

$$F_i' := \{C \vee y_0^{b(i)(0)} \vee \ldots \vee y_{c-1}^{b(i)(c-1)} \mid C \in F_i\}.$$

Using this, we define

$$join_Y(F_1, \ldots, F_l) := \bigcup_{i=1}^{l} F_i'.$$

Informally $join_Y(F_1, \ldots, F_l)$ is the conjunction of all clauses of the formulas F_i, where each clause is marked with i encoded in the signs of the added y_j variables. Now we prove that $join_Y(F_1, \ldots, F_l)$ is at least as hard as the hardest F_i.

Lemma 3.39. *Let $F_1, \ldots, F_l$ be formulas using the variables $x_1, \ldots, x_n$ with $l = 2^c$. Let $Y = \{y_0, \ldots, y_{c-1}\}$ be a disjoint set of variables. Let $F := join_Y(F_1, \ldots, F_l)$.*

For semantic (negative, regular, tree-like, general) resolution the following relationships hold: $s(F) \geq \max_{i=1}^{l} s(F_i)$ and $w(F) \geq \max_{i=1}^{l} w(F_i)$, where $s(G)$ is the minimal size of any semantic (negative, regular, tree-like, general) refutation of the formula G, and $w(G)$ is the minimal width of any semantic (negative, regular, tree-like, general) refutation of the formula G.

Proof. Let $1 \leq i \leq l$. Take a refutation R of F with width w and size s. Note that $F\lceil_\alpha$, where α sets y_j to $1 - b(i)(j)$ (and no other variables), is exactly F_i. Hence by Corollary 1.8 on page 16 there is a refutation of F_i with width at most w and size at most s. □

Lemma 3.40. *For every assignment β there is an ordered resolution refutation R of $\mathrm{XOR}(f_\beta(\mathrm{PP}_G))$ with $|R| = O(p(n))$ for some polynomial p.*

Proof. During this proof we will abbreviate $u^{1-\beta(u)}$ (i.e., a variable occuring positively in PP_G) with ${}^\beta u$, and $u^{\beta(u)}$ with ${}^\beta \bar{u}$.

We first fix some topological ordering $u_1, \ldots, u_n$ of G. Now following this ordering we derive $\mathrm{XOR}({}^\beta u_i) = ({}^\beta u_i \vee {}^\beta \bar{u}_i') \wedge ({}^\beta \bar{u}_i \vee {}^\beta u_i')$ for every u_i. For a source u_i we already have this in the formula. For any other node we already have derived these clauses for each of its predecessors v_i, $i = 1, \ldots, k$, since these occur earlier in the topological ordering.

The formula consisting of the clauses ${}^\beta v_i$, $i = 1, \ldots, k$ and $\bigvee_{i=1}^{k} {}^\beta \bar{v}_i$ is clearly unsatisfiable. Thus its xorification is as well. Since it uses only $2k$ variables, there is by Observation 1.6 on page 15 an ordered refutation with a size of at most $2^{O(k)}$ compatible with the above ordering. Note that k is the in-degree of node u which is bounded by $O(\log |V|)$.

By appending one of the clauses C of $\mathrm{XOR}({}^{\beta}u_i)$ to $\mathrm{XOR}(\bigvee_{i=1}^{k} {}^{\beta}\bar{v}_i)$ we get a derivation of C, and by duplicating this using the other one, a derivation of $\mathrm{XOR}({}^{\beta}u_i)$. The resulting derivation does still obey the ordering and does not resolve on u_i or u_i'.

After deriving $\mathrm{XOR}({}^{\beta}u_n)$ we finish the refutation by resolving $\mathrm{XOR}({}^{\beta}u_n)$ with $\mathrm{XOR}({}^{\beta}\bar{u}_n) = ({}^{\beta}u_n \vee {}^{\beta}u_n') \wedge ({}^{\beta}\bar{u}_n \vee {}^{\beta}\bar{u}_n')$.

$$\dfrac{\dfrac{{}^{\beta}u_n \vee {}^{\beta}u_n' \quad {}^{\beta}u_n \vee {}^{\beta}\bar{u}_n'}{{}^{\beta}u_n}\, u_n' \qquad \dfrac{{}^{\beta}\bar{u}_n \vee {}^{\beta}\bar{u}_n' \quad {}^{\beta}\bar{u}_n \vee {}^{\beta}u_n'}{{}^{\beta}\bar{u}_n}\, u_n'}{\square}\, u_n$$

This proves the lemma. $\square$

Lemma 3.41. *Let $F_0, \ldots, F_{l-1}$ be formulas using the variables $x_1, \ldots, x_n$ with $l = 2^c$. Let $Y = \{y_0, \ldots, y_{c-1}\}$ be a disjoint set of variables. Assume that for some ordering of the variables there are ordered refutations of* $\mathrm{XOR}(F_i)$ *of polynomial size for each i. Then there is an ordered refutation of $F := \mathrm{XOR}(join_Y(F_0, \ldots, F_{l-1}))$, whose size is polynomial in n and l.*

Proof. Fix some $0 \le i < l$. Now fix one clause C_i from $\mathrm{XOR}(y_0^{b(i)(0)} \vee \ldots \vee y_{c-1}^{b(i)(c-1)})$. Now the set of clauses in F that contain C_i as a subclause is exactly $\mathrm{XOR}(F_i)$ if C_i is removed from each of these clauses. Hence we can derive C_i from these clauses using the ordered refutation of $\mathrm{XOR}(F_i)$. We do this for every choice of i and C_i.

We have now derived $\mathrm{XOR}(\bigwedge_{i=1}^{l-1} y_0^{b(i)(0)} \vee \ldots \vee y_{c-1}^{b(i)(c-1)})$ without eliminating any of the variables y_j. This formula is unsatisfiable and has $c = \log l$ variables. By Observation 1.6 on page 15 any ordered refutation of a formula with c variables has a size of at most $2^{O(c)}$. Hence there is an ordered refutation of this formula with a size polynomial in l. $\square$

Now we prove a lower bound for semantic resolution. In the following we will use the term α-refutation to denote a resolution refutation with the property that one of the clauses used in each resolution step is falsified by the assignment α.

Let G be a graph and α, β total assignments. Now consider an α-refutation R of $f_\beta(\mathrm{PP}_G)$. Let $\bar{e}(\alpha, \beta)$ be the set of nodes v with $\alpha(v) \neq \beta(v)$. Let G' be the induced subgraph on the nodes $\bar{e}(\alpha, \beta)$. For a clause C in R, let $zeros(C, \beta)$ be the set of variables that appear in C as $v^{\beta(v)}$. We call these literals β-negative. We will call the other literals β-positive. Note that the variables that occur β-positively in a clause of $f_\beta(\mathrm{PP}_G)$, occur positively in the corresponding clause of PP_G (and analogously for β-negative).

Lemma 3.42. *Let C be a clause in R, where one variable v occurs β-positively with $v \in \bar{e}(\alpha, \beta)$. Then all parents of v in G' occur β-negatively in C.*

Proof. We prove this by induction on the maximal length of a path between C and an axiom in the proof. The only axiom that contains v β-positively contains all parents of v β-negatively.

Now let C be the resolvent of the clauses C_1 and C_2. v occurs β-positively in C_1 or C_2. W.a.l.o.g. v occurs β-positively in C_1. By induction C_1 contains all G'-parents of v. Assume that C does not contain all of these. Then one of them is eliminated in the resolution step. We call this one u. Note that $u \in \bar{e}(\alpha, \beta)$, since it is a node in G'. Since u occurs β-negatively in C_1, it occurs β-positively in C_2. Now α satisfies C_1, since $v \in \bar{e}(\alpha, \beta)$ and v occurs β-positively in C_1. Since $u \in \bar{e}(\alpha, \beta)$ and u occurs β-positively in C_2, α satisfies C_2, too. Thus C_1 cannot be resolved with C_2 in the α-refutation R. Thus C contains all the G'-parents of v. $\square$

Lemma 3.43. *For any clause C in R, let $S_C := zeros(C, \beta) \cap \bar{e}(\alpha, \beta)$. If $\mathrm{cPeb}(G', \emptyset, S_C) = p$, then on the path from C to $\Box$ there is at least one clause with p β-negative literals.*

Proof. We prove this by induction on the length of the (shortest) path between C and $\Box$. For $C = \Box$ we have $S_C = \emptyset$, and there is nothing to prove.

Now assume C is some other clause. Then C is resolved with some clause D yielding E with a shorter path from E to $\Box$. If $S_E \supseteq S_C$, then we are done by the induction hypothesis and Observation 3.31 on page 51. Otherwise the eliminated variable v is in S_C, and thus it occurs β-negatively in C. Hence v occurs β-positively in D, and by Lemma 3.42 , D contains all the G'-parents of v β-negatively. Therefore E contains these as well. Therefore we can easily pebble from S_E to S_C, which proves, together with the induction hypothesis for E, the induction assertion for C. $\square$

Lemma 3.44. *Let G be a pyramid-like dag with n nodes, and let α and β be total assignments. Let G' be the induced subgraph on $\bar{e}(\alpha, \beta)$ as above. If G' contains a node v with $\mathrm{cPeb}(G', \emptyset, \{v\}) = p$ such that there is a path in G from v to the sink t, then any α-refutation R of $f_\beta(\mathrm{PP}_G)$ contains a clause with at least p β-negative literals.*

Proof. There is at least one axiom in $f_\beta(\mathrm{PP}_G)$ that contains v β-negatively (either a pebbling axiom of an antecessor of v—there is one if $v \neq t$ since there is a path from v to t—, or v's sink axiom). This axiom is used in R, since $f_\beta(\mathrm{PP}_G)$ is satisfiable without it (again since there is a path from v to t). Therefore, by Lemma 3.43 , R contains a clause with at least p β-negative literals. $\square$

Lemma 3.45. *For infinitely many n and any $n' \geq n$, there are assignments $\beta_1, \ldots, \beta_{n'} \in \{0,1\}^n$ and pyramid-like graphs $G_1, \ldots, G_{n'}$ of in-degree $O(\log n)$ with n nodes such that the following holds: For any (total) assignment α, there is an i such that any α-refutation of $f_{\beta_i}(\mathrm{PP}_{G_i})$ contains one clause with $\Omega(\sqrt{n})$ β_i-negative literals.*

Proof. Since only the settings of the n variables in PP_{G_i} are relevant, we identify total assignments which set these to the same values. And we identify each assignment to these variables with a bitstring of length n (where each variable is associated with one position on the bitstring).

Fix m sufficiently large with $8|m$ and let $n = \sum_{i=1}^m i$. Let $n' \geq n$. Now we choose $\beta_1, \ldots, \beta_{n'}$ randomly and independently from the uniform distribution on $\{0,1\}^n$. We choose $G_1, \ldots, G_{n'}$ from the distribution $Pyr(m,d)$ with $d > \log_{8/5} m$. Fix $\alpha \in \{0,1\}^n$. G'_i denotes, as above, the subgraph of G_i induced by $\bar{e}(\alpha, \beta_i)$.

First we show that, with high probability, there is one G'_i that satisfies the conditions of Lemma 3.33 on page 52. Fix $1 \leq i \leq n'$. Now layer j of G'_i is expected to contain $j/2$ nodes. By Chernoff's bound, the probability that it contains fewer than $j/4$ nodes is less than $\exp(-j/16)$. The probability that any of the $m/8$ layers from m to $\frac{7}{8}m+1$ has less than $j/4$ nodes is at most

$$\sum_{j=(7/8)m+1}^{m} \exp\left(-\frac{j}{16}\right) \leq \frac{m}{8} \cdot \exp\left(-\frac{7m}{128}\right).$$

Now we bound the probability that any subset of layer j of size $s \leq j/8$ is not expanding into layer $j+1$ for $j < m$. Fix some subset S_1 of size s from layer j and a subset S_2 of size s from layer $j+1$. S_1 is not expanding if every parent v of every node in S_1 is either in S_2 or not at all in G'_i. The probability for the first is at most $s/(j+1)$, the probability for the second is $1/2$. The probability that any of these things happens is at most $1/2 + s/(j+1) < 5/8$. Therefore the probability that layer j does not expand into layer $j+1$ is bounded by

$$\begin{aligned} & \sum_{s=1}^{j/8} \binom{j+1}{s}^2 \left(\frac{5}{8}\right)^{ds} \\ < \ & \sum_{s=1}^{j/8} \binom{j+1}{s}^2 \frac{1}{m^{5s}} \\ \leq \ & \sum_{s=1}^{j/8} \binom{j+1}{s}^2 \frac{1}{(j+1)^{5s-2}m^2} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{s=1}^{j/8} \frac{(j+1)^2 \cdot j^2 \cdot (j-1)^2 \cdot \ldots \cdot (j+2-s)^2}{(s!)^2 \cdot (j+1)^{5s-2} \cdot m^2} \\
&< \sum_{s=1}^{j/8} \frac{1}{(s!)^2 \cdot (j+1)^{3s-2} \cdot m^2} \\
&\leq \frac{j}{8} \frac{1}{(j+1) \cdot m^2} \\
&< \frac{1}{8 \cdot m^2} < \frac{1}{m^2}
\end{aligned}
$$

The first inequality follows from $d > 5 \log_{8/5} m$, the second one follows from $j+1 \leq m$. The probability that any of the $m/8$ layers from m to $\frac{7}{8}m+1$ is not expanding is thus less than $\frac{m}{8}\frac{1}{m^2} < \frac{1}{m}$.

If neither of these bad events happens, then G'_i contains by Lemma 3.33 on page 52 a node v with $\text{cPeb}(G'_i, \emptyset, \{v\}) \geq m/8$. We write $A(i, \alpha)$ to denote the event that Lemma 3.33 on page 52 cannot be applied to G'_i.

To apply Lemma 3.44 on page 56 to G_i, we need a node v on layer $\frac{7}{8}m+1$ in G'_i from which there is a path in G_i to the sink t. First we will calculate an upper bound for the probability that a node v on layer j does not have a path to t. We will call this probability $p(i, j)$. If there is no node from v to t, then either there is no child of v on layer $j-1$, or none of the children has a path to t. The probability for the former is

$$
\left(\frac{j-1}{j}\right)^{d(j-1)},
$$

and the probability of the latter is at most $p(i, j-1)$. Thus

$$
\begin{aligned}
p(i,j) &\leq p(i, j-1) + \left(\frac{j-1}{j}\right)^{d(j-1)} \\
&\leq p(i, j-1) + \frac{1}{e^d} \qquad (\text{remember } \left(1 - \frac{1}{x}\right)^x \leq e^{-1}) \\
&< \frac{m}{e^d}
\end{aligned}
$$

The last step follows from the fact that $p(i, 1) = 0$, which holds since t is the only node on layer 1. With $d > 5 \log_{8/5} m$, this is less than $1/m^4$.

$$
\begin{aligned}
\frac{m}{e^d} &< \frac{m}{\exp(5 \log_{8/5} m)} \\
&= \frac{m}{\exp(5 \log_{8/5} e \log_e m)} \\
&= \frac{m}{m^{5 \log_{8/5} e}} = m^{1 - 5 \log_{8/5} e} \\
&< \frac{1}{m^4} \qquad (\text{note } 1 - 5 \log_{8/5} e < -4)
\end{aligned}
$$

The probability that there is no node on layer $\frac{7}{8}m+1$ in G_i' is $(1/2)^{m/8}$. Therefore the probability that there is no node v on layer $\frac{7}{8}m+1$ in G_i' from which there is a path in G_i to the sink t is at most

$$\frac{1}{m^4} + \left(\frac{1}{2}\right)^{\frac{m}{8}} \le \frac{1}{m^3}.$$

Let us call the event that there is no node to apply Lemma 3.44 on page 56 $B(i,\alpha)$.

Then the event $\overline{A(i,\alpha) \cup B(i,\alpha)}$ implies that any α-refutation of $f_{\beta_i}(\mathrm{PP}_{G_i})$ contains a clause with $m/8 = \Omega(\sqrt{n})$ β_i-negative literals. The probability of $A(i,\alpha) \cup B(i,\alpha)$ is at most

$$\frac{m}{8} \cdot \exp\left(-\frac{7m}{128}\right) + \frac{1}{m} + \frac{1}{m^3} < \frac{3}{m}.$$

The probability that $A(i,\alpha) \cup B(i,\alpha)$ holds for all $1 \le i \le n'$ is thus at most $\left(\frac{3}{m}\right)^{n'}$. Therefore the probability that $A(i,\alpha) \cup B(i,\alpha)$ holds for all $1 \le i \le n'$ and all $\alpha \in \{0,1\}^n$ is at most $2^n \left(\frac{3}{m}\right)^{n'}$, which is smaller than 1. Thus there are $\beta_1, \dots, \beta_{n'}$ and $G_1, \dots, G_{n'}$ such that, for any α there is some i such that any α-refutation of $f_{\beta_i}(\mathrm{PP}_{G_i})$ has at least width $\Omega(\sqrt{n})$. □

Theorem 3.46. *Let $n = \sum_{i=1}^{m} i$ with sufficiently large m, and let $n' > n$ be a power of 2. There are $\beta_1, \dots, \beta_{n'} \in \{0,1\}^n$ and dags $G_1, \dots, G_{n'}$ of in-degree $O(\log n)$ with n nodes, such that $\mathrm{XOR}(join_Y(f_{\beta_1}(\mathrm{PP}_{G_1}), \dots, f_{\beta_{n'}}(\mathrm{PP}_{G_{n'}})))$ yields an exponential separation between ordered and semantic resolution.*

Proof. The polynomial upper bound for ordered resolution follows directly from Lemma 3.40 on page 54 and Lemma 3.41 on page 55.

The exponential lower bound follows from the width lower bound proven in Lemma 3.45 on page 57, which holds for the whole formula because of Lemma 3.39 on page 54. The width lower bound implies a size lower bound by Lemma 3.37 on page 53. □

3.10.2 Ordered Resolution does not Simulate Semantic Resolution

The separation ord $\not\geq$ sem is implied by neg $\not\leq$ reg proven in Section 3.2.1, neg $\leq$ sem, and ord $\leq$ reg.

3.11 Regular and Semantic Resolution

In this section we prove reg <> sem. That reg $\not\leq$ sem is implied by ord $\not\leq$ sem proven in Section 3.10.1 and ord $\leq$ reg. That reg $\not\geq$ sem is implied by neg $\not\leq$ reg proven in Section 3.2.1 and neg $\leq$ sem.

3.12 Semantic and General Resolution

Here we prove sem $<$ dag. That sem $\leq$ dag follows immediately from the definition. The separation follows from the fact that ord $\not\leq$ sem, proven in Section 3.10.1.

3.13 Negative and General Resolution

In this section we prove neg $<$ dag. That neg $\leq$ dag follows immediately from the definition.

Since every negative refutation is a semantic one (with the all-one-assignment), this follows immediately from sem $<$ dag proven in Section 3.12. This was first proven by Goerdt [18] and later improved by Buresh-Oppenheim et al. [10].

3.14 Negative and Ordered Resolution

In this section the incomparability of negative and ordered resolution is shown (neg $<>$ ord).

3.14.1 Negative Resolution is Separated from Ordered Resolution

Theorem 3.47. *neg $\not\leq$ ord*

Proof. That neg $\leq$ ord is impossible, otherwise we would have tree $<$ neg $\leq$ ord and therefore tree $\leq$ ord, which contradicts the incomparability of tree-like and ordered resolution. Thus negative resolution is separated from ordered resolution. □

A direct proof of this separation was given by Bonet et al. [8].

3.14.2 Ordered Resolution is Separated from Negative Resolution

The separation ord $\not\leq$ neg is implied by ord $\not\leq$ sem proven in Section 3.10.1 and neg $\leq$ sem.

Chapter 4

Linear Resolution

This chapter is dedicated to linear resolution, which seems to be the most mysterious refinement discussed in this work. At first glance, linear resolution might seem incomplete, but it is complete, and it is still unknown if even general resolution is stronger. In this chapter we will present and prove all we know about the relative strength of linear resolution. Additionally we will introduce a modified version of linear resolution, which is as strong as general resolution, and use this to prove a necessary and sufficient condition for the equivalence of linear and general resolution.

4.1 Linear Resolution Simulates Tree-like Resolution

Here we show that linear resolution simulates tree-like resolution. The separation is proven later in Section 4.4. The following proof is due to Johannsen [24].

Theorem 4.1. *There is a linear resolution proof for an unsatisfiable formula F of size at most $2m$ if there is a tree-like resolution proof of size m.*

Since we get with Theorem 1.9 on page 16 from every tree-like resolution refutation R' a regular tree-like resolution refutation R with $|R| \leq |R'|$, the theorem follows from the following lemma.

Lemma 4.2. *If there is a regular tree-like derivation R of C from a set of clauses Γ, and no literal in C is eliminated in R, then there is a linear derivation of C from Γ with a size of at most $2|R|$.*

Note that all subtrees of a regular tree-like resolution refutation satisfy the second assumption.

Proof. We prove this by induction on the length of the proof. The base case is trivial since the derivation is already linear. Now let R end with the step

$$\frac{C \vee x \quad D \vee \bar{x}}{C \vee D},$$

where $C \vee x$ is derived from Γ through R_1 and $D \vee \bar{x}$ from $C_1, \dots, C_k \in \Gamma$ through R_2, and $|R| = |R_1| + |R_2| + 1$.

By the induction hypothesis, there is a linear derivation R_1' of $C \vee x$ from Γ with $|R_1'| \leq 2|R_1|$. The literal $\bar{x}$ occurs in at least one of the C_i, w.a.l.o.g. $C_1 = C_1' \vee \bar{x}$. Resolving $C \vee x$ with C_1 yields $C_1' \vee C$.

By replacing C_1 with $C_1' \vee C$ (and changing all descendants to keep it a derivation), we transform R_2 to a regular tree-like derivation R_2'' of $C \vee D$ (or $C \vee D \vee \bar{x}$ if there is a clause C_i with $i \neq 1$ that contains $\bar{x}$).

Since $|R_2| = |R_2''|$, and by the induction hypothesis, there is a linear derivation R_2' of $C \vee D$ (or $C \vee D \vee \bar{x}$) from the clauses $C_1' \vee C, C_2, \dots, C_k$ with $|R_2'| \leq 2|R_2|$. By appending R_2' to the end of R_1', we get a linear derivation of $C \vee D$ or $C \vee D \vee \bar{x}$. In the latter case we add an additional resolution step ($C \vee x$ resolved with $C \vee D \vee \bar{x}$ yields $C \vee D$). This linear derivation R' has size $|R'| \leq |R_1'| + |R_2'| + 2 \leq 2|R_1| + 2|R_2| + 2 = 2|R|$. $\square$

4.2 Linear Resolution with Restarts

Linear resolution with restarts [11] allows to continue the proof with a clause from the formula instead of the result of a resolution step.

We will call the linear parts of the proof *chains*, i.e., the part of the dag where the result of a resolution is immediately used to continue the proof. And we will call the step from one chain to the next *restart*, i.e., every C_i, C_{i+1} where C_{i+1} is not derived from C_i.

We will prove in this section that linear resolution with restarts is equivalent to general resolution. This is, as far as I know, a new result.

Theorem 4.3. *Linear resolution with restarts simulates general resolution.*

Proof. Let P be a resolution proof of length s of a formula F. Sort the clauses D_i in P by the length of the longest path in P between a clause from F and D_i. Then $D_s = \square$.

We now prove by induction that every clause in P can be derived with linear resolution with restarts by adding at most s clauses to a derivation P' of the previous clauses.

The beginning is clear ($D_1 \in F$).

Now we have to derive D_i. We already have a derivation P' that derives all D_j with $j < i$. All clauses used in P to derive D_i are already in P' since the longest path from an axiom to these is shorter than the longest path to D_i. Thus any path in P from an axiom to D_i appended to P' forms a linear derivation with restarts of D_i. Since there are only s clauses in P, we add at most s clauses to P'. $\square$

Since every linear resolution with restarts proof is also a general resolution proof, the latter obviously simulates the first.

Corollary 4.4. *Linear resolution with restarts and general resolution are equivalent.*

4.3 Linear Equivalent to General Resolution?

The question if linear resolution is separated from general resolution is still open. There is a proof for lin < dag in a paper by Buresh-Oppenheim et al. [10], but this proof is not correct. They use the proposition that a linear refutation R of F becomes a linear refutation of $F\lceil_\alpha$ by applying α to all clauses in R and removing some clauses. Unfortunately this proposition is not a fact.

4.3.1 A Necessary and Sufficient Condition

We now prove that linear resolution is equivalent to general resolution if a weaker version of Corollary 1.8 on page 16 holds for linear resolution. The following theorem and its proof are due to Hoffmann [21].

Hypothesis 4.5. *If there is a linear resolution proof R of size s for an unsatisfiable formula F, then given a partial assignment α there is a linear resolution proof R' for $F\lceil_\alpha =: F'$ of size at most $p(s)$ for some polynomial p.*

Theorem 4.6. *If Hypothesis 4.5 holds, then linear resolution simulates general resolution.*

Proof. Let R be a resolution refutation of F. By Theorem 4.3 on the facing page we get a linear refutation with restarts R' that is at most polynomially bigger.

Now we construct F' by adding clauses to F. For each restart C_i, C_{i+1} we add the clauses $C_{i+1} \vee \bar{y}_i$ and $\bar{a} \vee y_i$ for all literals $a \in C_i$ and some new y_i.

By using these clauses to derive C_{i+1} from C_i, we get a linear refutation (without restarts) R'' of F'. $|R''| \leq l \cdot |R'|$ where l is the width of R'.

Since $F = F'\lceil_\alpha$ where α sets all y_i to 1 and no other variables, there is by Hypothesis 4.5 a linear refutation $\widetilde{R}$ of F with a size of at most $p(|R''|)$ for some polynomial p.

With the above inequalities we have $|\widetilde{R}| \leq q(|R|)$ for some polynomial q and this proves the theorem. $\square$

If linear resolution simulates general resolution, then Hypothesis 4.5 on the preceding page holds, since the size of general resolution refutations is preserved under restrictions.

Corollary 4.7. *Hypothesis 4.5 on the previous page is equivalent to dag = lin.*

4.3.2 Simulation on Special Formulas

We show that linear resolution simulates general resolution if we add special tautologic clauses. Note that the number of added clauses is only $O(n^2)$.

Definition 4.8.

$$\mathrm{ADDTAUT}(F) := F \cup \tilde{F}$$

where $\tilde{F} := \{x \vee \bar{x} \vee y^{\varepsilon} \mid x, y \in \mathrm{var}(F), \varepsilon \in \{0, 1\}\}$

Lemma 4.9. *If there is a general resolution refutation R of F, then there is a linear resolution of* ADDTAUT(F) *that is at most polynomially bigger than R. Linear resolution simulates general resolution on formulas of the form* ADDTAUT(F)*.*

While the above lemma is due to Buresh-Oppenheim and Pitassi [11], the following proof is different from theirs.

Proof. By Theorem 4.3 on page 62 we can get a linear refutation R' with restarts of F that is at most polynomially bigger than R. Clearly this is also a refutation of ADDTAUT(F). We construct from this a linear refutation R'' by removing all restarts.

Let C_i, C_{i+1} be any restart, i.e., C_{i+1} is an axiom and not derived from C_i. Let x be some variable occurring in C_{i+1}. Now we resolve C_i with $x \vee \bar{x} \vee l$ for every literal $l \in C_i$. We obtain $x \vee \bar{x}$. By resolving this with the axiom C_{i+1}, we obtain C_{i+1} (note that C_{i+1} contains the variable x). From there we continue the proof as before. We repeat this step until there are no restarts left.

There are at most $|R'|$ restarts and we add at most as many clauses per restart as the width of R', therefore R'' is at most polynomially bigger than R' and thus R.

The second part of the lemma follows from the fact that the additional tautological clauses do not shorten any general resolution proof (Theorem 1.12 on page 19). □

4.4 Linear Resolution is not Simulated by . . .

Now we prove that linear resolution is separated from tree-like, regular, ordered, semantic, and negative resolution. This is due to Buresh-Oppenheim and Pitassi [11].

Theorem 4.10. *Linear resolution is separated from every resolution refinement S_{ref} from which general resolution is separated and that does not have smaller proofs for* ADDTAUT(F) *than for* F.

Proof. Let F_n be a family of formulas separating dag from S_{ref}. Then ADDTAUT(F_n) separates lin from S_{ref}.

Assume this is not the case. Then we can construct, starting with a short general proof of F_n, a short linear proof of ADDTAUT(F_n) by Lemma 4.9 on the facing page. From this we construct a short S_{ref} proof of ADDTAUT(F_n) using the assumption. Then there is also an S_{ref} proof of the same size for F_n, which contradicts the separation of dag from S_{ref}, since the overall blowup is only polynomial. $\square$

The above theorem holds for tree-like, regular, ordered, semantic, and negative resolution.

Chapter 5

Lower Bounds for DLL

In this chapter we will first present and prove the well-known connection between resolution and DLL. Then we will use this connection to prove lower bounds for DLL on satisfiable formulas following a work by Alekhnovich et al. [1].

5.1 On Unsatisfiable Formulas

Lower bounds for DLL on unsatisfiable formulas usually result from the following connection between DLL and resolution.

Theorem 5.1. *If a DLL algorithm A needs s calls of* DLL *to prove the unsatisfiability of some formula F, there is a (tree-like) resolution proof of size s.*

Proof. We prove this by labeling the call tree T of a run of A on F in such a way that we get a resolution refutation of F.

Each leaf is labeled with a clause falsified by the setting at the leaf.

An inner node is labeled with the clause resulting from the resolution of the clauses of its children on the decision variable v if both clauses contain v. Otherwise it is set to one of the clauses that does not contain v.

It is obvious that it is always possible to label T in such a way. It is also obvious that the resulting tree is a resolution derivation of the clause labeling the root. We only need to prove that the root is labeled with the empty clause.

We prove the following claim, which implies the above, by induction on the steps: The clause labeling a node is falsified by the partial assignment of the node. Since the assignment of the root node does not assign any variable to a value, it is labeled with the empty clause.

For the leaves this follows directly from the construction.

Suppose the clauses labeling the children of an inner node already fulfill this. Then, if one of these clauses is used to label the current node, it does not contain the decision variable, which is the only variable set on the child nodes but not on the current one. So it is falsified by the assignment of the current node. If the clause is derived by resolution, the only literal in the clauses on both children that is not set by the assignment of the current node is removed by the resolution. □

A similar connection between general resolution and DLL with learning and restarts was shown by Beame et al. [4]. There is currently work in progress to show if there is a similar connection between DLL with learning but without restarts and regular tree-like resolution with lemmas [22].

5.2 On Satisfiable Formulas

If **NP** = co-**NP**, a DLL algorithm with a good[1] heuristic would need only polynomial time to find a satisfying assignment. Since we do not know wether **NP** = co-**NP** holds or not, we restrict the heuristic in a certain way and give lower bounds for DLL algorithms using such a restricted heuristic.

5.2.1 Drunken Heuristic

A *drunken heuristic* has no restrictions on how to choose a variable, but the value to which the selected variable is assigned is chosen randomly (independently and uniformly), i.e., the heuristic chooses only the variable (without knowing the value). The algorithm is shown in Figure 5.1 on the next page.

The following lower bound and its proof are due to Alekhnovich et al. [1].

Definition 5.2. *Let G_n be a family of unsatisfiable formulas with n variables $x_1, \ldots, x_n$ that require tree-like refutations with exponential size (e.g., the formula* PHP *from Section 2.1). We will write $G_n^{(j)}$ for a copy of the formula G_n where each variable x_i is replaced by $x_i^{(j)}$. We define*

$$\widetilde{G}_n^{(j)} := (G_n^{(j)} \vee x_1^{(j)}) \wedge \ldots \wedge (G_n^{(j)} \vee x_n^{(j)}).$$

We consider $G_n^{(j)} \vee x_1^{(j)}$ as CNF, i.e., $x_1^{(j)}$ is added to all clauses of $G_n^{(j)}$ and all clauses containing $\neg x_1^{(j)}$ are removed. Finally we define

$$H_n := \widetilde{G}_n^{(1)} \wedge \ldots \wedge \widetilde{G}_n^{(n)}.$$

Note that H_n has a size polynomial in n (and the size of G_n) and n^2 variables. H_n is satisfiable only by the assignment that assigns all variables to 1.

[1] A "good" heuristic could in this case just calculate some satisfying assignment α in polynomial time and then choose the variables in any order and their values according to α.

```
DrunkenDLL(F, α)
    if F⌈_α = 1
        return α
    if F⌈_α = 0
        return UNSAT

    (v) := HEUR(F, α)
    # HEUR is the heuristic that selects
    # the variable v to be set next.
    # v is called decision variable.

    ε := RANDOM ∈ {0, 1}

    σ := DrunkenDLL(F, α ∪ {v ↦ ε})

    if σ ≠ UNSAT
        return σ
    else
        return DrunkenDLL(F, α ∪ {v ↦ ¬ε})
```

Figure 5.1: Drunken DLL Algorithm

Lemma 5.3. *Let F and G be formulas with disjoint variables. If F is satisfiable and G unsatisfiable, then the smallest refutation of $F \wedge G$ is as big as the smallest refutation of G.*

Proof. Every refutation of G is also a refutation of $F \wedge G$. But we cannot use any clause of F in a refutation of $F \wedge G$, since we cannot resolve any clause derivable from F with any derivable from G, and we cannot derive $\square$ from F. Thus every refutation of $F \wedge G$ is also one of G. $\square$

Lemma 5.4. *The smallest refutation of $\widetilde{G}_n^{(j)}\lceil_{x_i^{(j)}:=0}$ is at most polynomially smaller than the smallest one of $G_n^{(j)}\lceil_{x_i^{(j)}:=0}$.*

Proof. Note that

$$\widetilde{G}_n^{(j)}\lceil_{x_i^{(j)}:=0} = G_n^{(j)}\lceil_{x_i^{(j)}:=0} \wedge \bigwedge_{l=1, l\neq i}^{n} \left(G_n^{(j)}\lceil_{x_i^{(j)}:=0} \vee x_l^{(j)} \right)$$

Further note that all clauses appearing in the big disjunction either appear in $G_n^{(j)}\lceil_{x_i^{(j)}:=0}$ or they are subsumed by some clause therein. Thus the lemma follows from Theorem 1.11 on page 18. $\square$

Theorem 5.5. *The probability that a drunken DLL run on H_n needs less than an exponential number of steps is at most 2^{-n}.*

Proof. Consider the case where the first variable occuring in $\widetilde{G}_n^{(j)}$ is set. With probability $\frac{1}{2}$ it is set to 0, resulting in an unsatisfiable formula. The smallest resolution refutation of this formula has, by Lemma 5.3, Lemma 5.4 and the definition of H_n, a size exponential in n. By Theorem 5.1 on page 67 the recursive call of DrunkenDLL with this partial assignment needs at least an exponential number of recursive calls before it returns UNSATISFIABLE.

Since there are n subformulas $\widetilde{G}_n^{(j)}$, the probability that this does not happen is 2^{-n}. □

5.2.2 Myopic Algorithms

A myopic algorithm uses a myopic heuristic, i.e., one that cannot use the whole formula at once. Here a *myopic heuristic* may use the following information:

- the whole formula with all negation signs removed
- the number of occurrences of each literal
- $K(n)$ clauses of the formula, where n is the number of the variables in the original formula and $K(n) := n^{1-\varepsilon}$ with $\varepsilon > 0$
- information revealed in calls upward in the call stack

While a myopic heuristic is generally unable to read all clauses that a variable occurs in, in the formulas we will use every variable occurs in at most $O(\log n)$ clauses.

We will construct formulas from special expander graphs, following a work by Alekhnovich et al. [1]. We need some additional definitions and lemmas.

But first we change our definition of formula slightly. In this section we will consider formulas as multisets instead of sets, i.e., a formula might contain a clause multiple times. This is probably more realistic, since it is not useful in a real implementation to remove duplicate clauses from the intermediate formulas during the run. And it is useful for the proof.

An expander is a graph with bounded degree and with the property that each subset of nodes has many neighbours. We will identify a graph with its adjacency matrix and define expanders in terms of $m \times n$ matrices over $\{0, 1\}$. Note that we will identify rows and columns with their numbers.

Definition 5.6. *For some vector $v = (v_1, \ldots, v_m)$ and a set $I \subseteq [m]$, we will write v_I for the subvector $(v_{i_1}, \ldots, v_{i_{|I|}})$ with $i_1 < \ldots < i_{|I|} \in I$.*

Similarly, for an $m \times n$ matrix A and $I \subseteq [m]$, we will write A_I to denote the submatrix consisting of the rows I. In particular we will use A_i for $A_{\{i\}}$ and identify this with the set $\{j \mid A_{ij} = 1\}$.

Let A be an $m \times n$ matrix and $I \subseteq [m]$ a set of rows. We define the boundary *$\partial_A I$ (or ∂I) of I as the set*

$$\{j \in [n] \mid \text{there is exactly one row } i \in I \text{ with } j \in A_i\}.$$

The elements of $\partial_A I$ are called boundary elements.

Definition 5.7. *An $m \times n$ matrix A is called (r, s, c)-*boundary expander *if*

1. *$|A_i| \leq s$ for all $i \in [m]$, and*
2. *$\forall I \subseteq [m]$ $(|I| \leq r \Rightarrow |\partial I| \geq c \cdot |I|)$.*

Definition 5.8. *An $m \times n$ matrix A is called (r, s, c)-*expander *if*

1. *$|A_i| \leq s$ for all $i \in [m]$, and*
2. *$\forall I \subseteq [m]$ $(|I| \leq r \Rightarrow |\bigcup_{i \in I} A_i| \geq c \cdot |I|)$.*

A boundary expander requires, unlike an expander, the existence of unique neighbours. Although this is stronger, we have the following lemma.

Lemma 5.9 (Alekhnovich et al. [1])**.** *Every $(r, 3, c)$-expander is also an $(r, 3, 2c - 3)$-boundary expander.*

The following lemma simply states that the expanders we need do actually exist. It can be proven by showing that a certain probabilistic process generates such an expander. In fact this happens with high probability.

Lemma 5.10 (Alekhnovich et al. [1])**.** *For every sufficiently large n, there is an $n \times n$ matrix $A^{(n)}$ with full rank such that $A^{(n)}$ is an $(n/\log^{14} n, 3, 25/13)$-expander, and furthermore, for every column j of $A^{(n)}$ there are at most $O(\log n)$ rows i with $j \in A_i$.*

The following inference relation between sets of rows was introduced by Alekhnovich and Razborov [3].

Definition 5.11. *Let $A \in \{0, 1\}^{m \times n}$ be an $(r, 3, c)$-boundary expander. For a set of columns $J \subseteq [n]$, we define the inference relation $\vdash_J$ on subsets of rows as follows:*

$$I \vdash_J I_1 \Leftrightarrow |I_1| \leq \frac{r}{2} \wedge \partial_A(I_1) \subseteq \left[\bigcup_{i \in I} A_i \cup J\right]$$

Let the closure *$Cl(J)$ of J be the set of all rows which can be inferred via $\vdash_J^*$ from the empty set, i.e., $i \in Cl(J)$ if there is a set $\{I_1, \ldots, I_k\}$ with $i \in I_k$, $I_1 = \emptyset$, and $\bigcup_{l=1}^{j} I_l \vdash_J I_{j+1}$.*

Lemma 5.12 (Alekhnovich et al. [3]). *If* $|J| \leq \frac{cr}{2}$*, then* $|Cl(J)| \leq \frac{r}{2}$*.*

We need the following inference relation to extract a good expander from the matrix that corresponds to a partial assignment during the run of the myopic DLL algorithm.

Definition 5.13. *Let* $A \in \{0,1\}^{m \times n}$ *be an* $(r,3,c)$*-boundary expander. For a set of columns* $J \subseteq [n]$*, we define the inference relation* $\vdash'_J$ *on subsets of rows as follows:*

$$I \vdash'_J I_1 \Leftrightarrow |I_1| \leq \frac{r}{2} \wedge \left| \partial_A(I_1) \setminus \left[\bigcup_{i \in I} A_i \cup J \right] \right| < \frac{c}{2} |I_1|$$

For a set I *of rows and a set* J *of columns we define a* cleaning *step: If there is a nonempty set* I_1 *of rows such that* $I \vdash'_J I_1$*, then add* I_1 *to* I *and remove all rows in* I_1 *from* A*.*

Now fix some order on sets of rows, set $I = \emptyset$ *and repeat the cleaning step as long as it is applicable. We will call the content of* I *at the end* $Cl^e(J)$*.*

The following lemma shows how to use the above inference relation to extract an expander from a set of columns.

Lemma 5.14 (Alekhnovich et al. [1]). *Let* A *be a matrix as above and* J *be a set of columns. Let* $I' := Cl^e(J)$ *and* $J' := \bigcup_{i \in Cl^e(J)} A_i$*. Remove the rows in* I' *and the columns* J' *from* A *and call the result* $\widehat{A}$*. If* $\widehat{A}$ *is non-empty, then it is an* $(r/2, 3, c/2)$*-boundary expander.*

Lemma 5.15 (Alekhnovich et al. [1]). *If* $|J| < \frac{cr}{4}$*, then* $|Cl^e(J)| < \frac{2}{c}|J|$*.*

Lemma 5.16 (Alekhnovich et al. [1]). *Let* $A \in \{0,1\}^{m \times n}$ *be an* $(r,3,c')$*-expander,* $X = \{x_1, \ldots, x_n\}$ *be a set of variables,* $\widehat{X} \subseteq X$ *with* $|\widehat{X}| < r$*,* $b \in \{0,1\}^m$*, and* $L = (l_1, \ldots, l_k)$ *(with* $k < r$*) a tuple of equations from* $Ax = b$*. Let* $\mathcal{L}$ *be the set of assignments with domain* $\widehat{X}$ *that can be extended to assignments with domain* X *which satisfy* L *(in* $\mathbb{F}_2$*). If* $\mathcal{L}$ *is not empty, it is an affine subspace of* $\{0,1\}^{|\widehat{X}|}$ *of dimension greater than* $|\widehat{X}| \cdot \left(\frac{1}{2} - \frac{14 - 7c'}{2(2c'-3)} \right)$*.*

Now we define the formula $\Phi_A(b)$ and a special property of a partial assignment used in the remainder of this section.

Definition 5.17. *Let* A *be an* $(r,3,c)$*-boundary expander. Let* $b \in \{0,1\}^n$*. Then* $\Phi_A(b)$ *(also* $\Phi(b)$*) is the formula (in CNF) using the variables* $x_1, \ldots, x_n$ *that states* $Ax = b$ *(in* $\mathbb{F}_2$*), where* $x = (x_1, \ldots, x_n)$*. For each equation* $a_{ij_1} x_{j_1} + a_{ij_2} x_{j_2} + a_{ij_3} x_{j_3} = b_i$*, there are up to 4 clauses in* $\Phi(b)$*. Note that there are at most three 1s in a row, since* A *is an* $(r,3,c)$*-boundary expander.*

We will, in a slight abuse of notation, identify the variable x_j *with its column* j*.*

$\Phi(b)$ has some properties which are very useful in the proof. First, there is exactly one satisfying assignment, since A has full rank. Second, every variable occurs positively as often as negatively in non-unit clauses (here we use the fact that the formula is a multiset of clauses).

Definition 5.18. *We call a partial assignment α* locally consistent *in respect to $Ax = b$ iff α can be extended to a total assignment which satisfies the equations corresponding to $Cl(J)$ where J is the domain of α:*

$$A_{Cl(J)}x = b_{Cl(J)}$$

Lemma 5.19. *Let A be an $(r, 3, c)$-boundary expander, $b \in \{0, 1\}^m$ and α be a locally consistent assignment. Then, for any set $I \subset [m]$ with $|I| \leq r/2$, α can be extended to an assignment β which satisfies $A_I x = b_I$.*

Proof. Let J be the domain of α.

We prove this by contradiction. Assume there is a set I such that α cannot be extended to satisfy $A_I x = b_I$. We choose a minimal I with this property. All boundary variables of I must be assigned to a value by α, short $\partial_A(I) \subseteq J$, otherwise we could remove the equation with the unset boundary variable, which would contradict the minimality of I. Therefore $Cl(J) \supseteq I$, which contradicts the local consistency of α. $\Box$

We prove that certain unsatisfiable formulas resulting from the above construction and a partial assignment need tree-like refutations of expontial size.

Lemma 5.20. *For an $(r, 3, c)$-boundary expander A and a vector $b \notin \mathrm{Im}(A)$, every resolution refutation of $\Phi_A(b)$ has a width of at least $cr/2$.*

Proof. For a clause C we define

$$\mu(C) := \min_{(A_I x = b_I) \models C} |I|.$$

This is subadditive. For any clause C appearing in $\Phi_A(b)$ we have $\mu(C) = 1$. We have $\mu(\Box) \geq r$ for the following reason: Assume $\mu(\Box) < r$. Then there is a set I with $|I| < r$ and $(A_I x = b_I) \models \Box$. Any set I with $|I| < r$ has at least one boundary variable. Let I' be I with one boundary element removed. Then $(A_{I'} x = b_{I'}) \models \Box$. By repeating this we get $(A_\emptyset x = b_\emptyset) \models \Box$. But $A_\emptyset x = b_\emptyset$ is satisfiable.

Therefore every resolution refutation of $\Phi_A(b)$ contains a clause C with $\frac{r}{2} \leq \mu(C) < r$. Now take a minimal set I such that $(A_I x = b_I) \models C$. C has to contain all variables in $\partial_A(I)$. Otherwise let $A_i x = b_i$ be the equation that contains the boundary variable not in C. Then $(A_{I \setminus \{i\}} x = b_{I \setminus \{i\}}) \models C$, which contradicts the minimality of I. There are at least $c \cdot |I| \geq cr/2$ elements in $\partial_A(I)$, all of these appear in C, thus C has at least width $cr/2$. $\Box$

Theorem 5.21. *If a locally consistent partial assignment α that assigns at most $cr/4$ variables to a value results in an unsatisfiable formula $\Phi(b)\lceil_\alpha$, then every tree-like resolution refutation of $\Phi(b)\lceil_\alpha$ has size $2^{\Omega(r)}$.*

Proof. Let V be the domain of α, $I := Cl^e(V)$ and $J := \bigcup_{i \in I} A_i$. By Lemma 5.15 on page 72, $|I| \leq r/2$. Therefore by Lemma 5.19 on the previous page we can extend α to a partial assignment β such that β assigns all variables in J to a value and satisfies $A_I x = b_I$.

The formula $\Phi(b)\lceil_\beta$, which is still unsatisfiable, is the encoding of a linear equation system $A'x = b'$ where A' results from A by removing all rows in I and all columns in J. By Lemma 5.14 on page 72, A' is an $(r/2, 3, c/2)$-boundary expander. The minimal width in Lemma 5.20 on the previous page and Corollary 2.10 on page 28 yield the lower bound in the theorem. □

Next we will prove the lower bound for myopic algorithms on $\Phi_A(b)$ using an $n \times n$ expander A provided by Lemma 5.10 on page 71. Let $r := n/\log^{14} n$, $c' := 25/13$ and $c = 2c' - 3$. Thus by Lemma 5.9 on page 71, A is an $(r, 3, c)$-boundary expander. We will prove the lower bound for a *clever myopic* algorithm. We call a myopic algorithm clever if it

- has the ability to read all clauses in $Cl(J)$ for free if it reveals at least one occurance of each variable in J.
- selects one of the revealed variables.
- never makes a stupid move: Whenever it reveals the clauses D and chooses the variable x_j, it does assign it to the correct value if D implies such a value.

The proof works by showing that the algorithm cannot get enough information about the formula during the first steps and thus needs to refute a formula which is hard to refute.

Lemma 5.22. *After the first $\left\lfloor \frac{cr}{6K} \right\rfloor$ steps a clever myopic algorithm knows at most $r/2$ bits of b.*

Proof. At each step the algorithm reads K clauses, and thus at most $3K$ different variables. After $\left\lfloor \frac{cr}{6K} \right\rfloor$ steps these are at most $cr/2$ variables, thus by Lemma 5.12 on page 72, the algorithm can know at most the clauses for $r/2$ of the equations, and thus at most $r/2$ bits of b. □

Lemma 5.23. *During the first $\left\lfloor \frac{cr}{6K} \right\rfloor$ steps the current partial assignment made by a clever myopic algorithm is locally consistent (in particular, the algorithm does not backtrack).*

Proof. We prove this by induction on the number of steps. The first assignment is empty and thus locally consistent. A clever myopic algorithm will always extend a locally consistent assignment by definition in such a way that it is still locally consistent if this is possible. By Lemma 5.19 on page 73 this is possible as long as $|Cl(J \cup \{x_j\})| \leq r/2$, where J is the domain of the current assignment and x_j is the variable chosen in this step. This is the case during the first $\lfloor \frac{cr}{6K} \rfloor$ steps. □

Now we are ready to prove the main theorem of this section, the lower bound for myopic algorithms.

Theorem 5.24. *Let b be chosen randomly (uniformly and independently) from $\{0,1\}^n$. Then every deterministic (clever) myopic DLL algorithm $\mathcal{A}$, that reads at most $K = K(n)$ clauses per recursive call, needs $2^{\Omega(r)}$ recursive calls to refute $\Phi(b)$ with probability $2^{-\Omega(r/K)}$.*

Proof. Everytime $\mathcal{A}$ reads the clauses corresponding to one equation $A_i x = b_i$, it learns one bit of b. After the first $t := \lfloor \frac{cr}{6K} \rfloor$ steps it has learned (by Lemma 5.22 on the preceding page) at most $r/2$ bits of b. Let I_t be the revealed bits and let R_t be the set of the t variables (note that by Lemma 5.23 on the facing page, $\mathcal{A}$ did not backtrack until now) which are assigned to a value at this time. We will call the current partial assignment α_t. Let $E := [(A^{-1}b)_{R_t} = \alpha_t]$ be the event that α_t assigns all variables in R_t to the correct value. We identify here the partial assignment with a bitvector of length t such that the above works. This event is equivalent with "$\Phi(b)\lceil_{\alpha_t}$ is satisfiable". Now we want to estimate the conditional probability

$$\Pr[E \mid I_t = I, R_t = R, b_{I_t} = \delta, \alpha_t = \alpha],$$

for some $I \subset [n]$, $R \subset [n]$, $\delta \in \{0,1\}^{|I|}$ and $\alpha \in \{0,1\}^R$. If this conditional probability is small (for all I, R, δ, α), then the probability of E is small.

We now use Lemma 5.16 on page 72 and set $L = \{A_i x = \delta_i\}_{i \in I}$, X to the set of variables in L and $\widehat{X} = R$. Then $\dim \mathcal{L} > \frac{2}{11}|R|$, where $\mathcal{L}$ is the set of locally consistent assignments with domain R. Define

$$\hat{b}_i = \begin{cases} \delta_i & i \in I \\ b_i & \text{otherwise.} \end{cases}$$

Note that $\hat{b}$ has the distribution of b when we fix $I_t = I$ and $b_I = \delta$. The vector $\hat{b}$ is independent of the event $E_1 := [I_t = I \wedge R_t = R \wedge b_{I_t} = \delta \wedge \alpha_t = \alpha]$, since we only need to look at the bits b_I to check if E_1 holds. $(A^{-1}\hat{b})_R$ is distributed uniformly on $\mathcal{L}$, thus

$$\begin{aligned} & \Pr[E \mid I_t = I, R_t = R, b_{I_t} = \delta, \alpha_t = \alpha] \\ = \; & \Pr[(A^{-1}\hat{b})_R = \alpha \mid I_t = I, R_t = R, b_{I_t} = \delta, \alpha_t = \alpha] \\ = \; & \Pr[(A^{-1}\hat{b})_R = \alpha] \\ \leq \; & 2^{-\dim \mathcal{L}} \end{aligned}$$

$$
\begin{aligned}
&< 2^{-\frac{2}{11}|R|} \\
&\leq 2^{-\frac{cr}{1000K}}
\end{aligned}
$$

Since by Lemma 5.23 on page 74 α_t is locally consistent, it takes $2^{\Omega(r)}$ steps to refute the formula that results if E does not happen (by Theorem 5.21 on page 74 and Theorem 5.1 on page 67). □

Corollary 5.25. *Let b be chosen randomly (uniformly and independently) from $\{0,1\}^n$, and choose enough random bits for the algorithm in the same way. Then every (randomized) (clever) myopic DLL algorithm, that reads at most $K = K(n)$ clauses per recursive call, needs $2^{\Omega(n \log^{-14} n)}$ recursive calls to refute $\Phi(b)$ with probability $2^{-\Omega(K^{-1} n \log^{-14} n)}$.*

Chapter 6

Conclusion

We have studied several refinements of resolution, in particular tree-like, regular, ordered, negative, semantic, and linear resolution as well as regular tree-like resolution with lemmas. Additionally, we presented some results concerning the connection between resolution and DLL, the basis for most complete **SAT**-solvers.

We gave complete proofs for all known simulations and separations of the different refinements. Only one well-known result about graphs was just quoted. While proofs for most of the results can be found in the given literature, the proofs compiled in this work share a common notation and do in fact prove what is needed. Some of them are simpler than the version in the literature and some errors were corrected.

Furthermore we introduced and studied a new approach to learn more about the strength of linear resolution. This approach leads us to a new and more elegant proof of the result which is the base of the proof of most separations between linear resolution and the other refinements.

Open Questions

There are still some open questions regarding the relative strengths of resolution refinements. Linear resolution is still somewhat mysterious. It is still unknown if it simulates any of the other refinements (except tree-like resolution). In particular there is still no known separation between general and linear resolution. The influence of weakening on the size of linear resolution refutations also is not yet clear. While it is obvious that linear resolution refutations are not preserved under restrictions, whether there are not much bigger[1] refutations for restricted formulas is still unresolved.

The other source of open questions is regular tree-like resolution with lemmas. It is not clear if it simulates linear, semantic, or negative resolution. We also do not know whether the separation is between rtrl and

[1] See Hypothesis 4.5 on page 63 for the exact meaning of "not much bigger".

general resolution, between rtrl and regular resolution, or if there are both separations. Similar to linear resolution, the influence of weakening and restrictions is unknown. For regular tree-like resolution with lemmas there is currently work in progress [22].

Bibliography

[1] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. *J. Autom. Reasoning*, 35(1-3):51–72, 2005. 67, 68, 70, 71, 72

[2] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing (accepted for publication)*, 2007. Preliminary version in STOC 2002. 40, 43, 46

[3] Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. In *42nd Annual Symposium on Foundations of Computer Science (FOCS 2001)*, pages 190–199, Las Vegas, Nevada, USA, October 2001. IEEE Computer Society. 71, 72

[4] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res. (JAIR)*, 22:319–351, 2004. 68

[5] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, pages 274–282, Burlington, Vermont, USA, October 1996. IEEE Computer Society. 23

[6] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004. 32, 34

[7] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001. 27, 29

[8] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000. 35, 36, 60

[9] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001. 41, 48

[10] Josh Buresh-Oppenheim, David Mitchell, and Toniann Pitassi. Linear and negative resolution are weaker than resolution. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(074), 2001. 49, 60, 63

[11] Josh Buresh-Oppenheim and Toniann Pitassi. The complexity of resolution refinements. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, pages 138–. IEEE Computer Society, 2003. 48, 51, 62, 64

[12] Samuel R. Buss. An introduction to proof theory. In *Handbook of Proof Theory*, chapter 1. Elsevier, Amsterdam, 1998. 15

[13] Samuel R. Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In Mogens Nielsen and Wolfgang Thomas, editors, *CSL*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156. Springer, 1997. 24

[14] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979. 11, 12

[15] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. 19

[16] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960. 19

[17] Andreas Goerdt. Davis-Putnam resolution versus unrestricted resolution. *Ann. Math. Artif. Intell.*, 6(1-3):169–184, 1992. 47

[18] Andreas Goerdt. Unrestricted resolution versus N-resolution. *Theor. Comput. Sci.*, 93(1):159–167, 1992. 60

[19] Andreas Goerdt. Regular resolution versus unrestricted resolution. *SIAM J. Comput.*, 22(4):661–683, 1993. 46

[20] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985. 23

[21] Jan Hoffmann. Personal communication, 2007. 63

[22] Jan Hoffmann. Resolution proofs and DLL algorithms (working title). Diplomarbeit, LMU München, 2007. In preparation. 48, 68, 78

[23] Jan Johannsen. Exponential incomparability of tree-like and ordered resolution. Unpublished Draft, 2001. http://www.tcs.informatik.uni-muenchen.de/~jjohanns/notes/string.ps.gz. 35

[24] Jan Johannsen. Unpublished, 2005. 61

[25] Jan Johannsen. Personal communication, 2007. 46

[26] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977. 22

[27] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 128–136, San Francisco, CA, USA, January 2000. ACM/SIAM. 33

[28] John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965. 9

Index