Project Report On Resolution Proof System

ANIL SHUKLA

THEORETICAL COMPUTER SCIENCE

THE INSTITUTE OF MATHEMATICAL SCIENCES

JUNE 2013

Dedicated To My Teachers And Parents

Acknowledgements

I would like to thank Prof. Meena Mahajan for her guidance and patience in this journey of preparing and writing the report. Without her the report would not have existed. The best part about this journey is the freedom that she has given to me.

Also, while preparing this report, the presentation sessions with her and Ramanathan S. Thinniyam has helped me alot in developing my understanding about the subject.

Finally, I would like to thank my friends Anish Mallick and Raja S. for useful discussions.

Anil Shukla

Theoretical Computer Science

The Institute of Mathematical Sciences

ABSTRACT

A propositional proof system (*pps*) is a proof system for $TAUT$. More formally, a *pps* is a polynomial time computable function f whose range is the set of all propositional tautologies.

In this report we investigate a particular *pps*, "resolution proof system". We review the first exponential size lower bound for resolution, proved by Haken in 1985 [16].

We review relationships among the three complexity measures of resolution: size, width and space. In particular, the report includes size-width relation for general and tree-like resolutions (by Ben-sasson and Wigderson [9]), size-space relation for tree-like resolutions (by Esteban and Torán [12]), and width-space relation for general resolutions (by Atserias and Dalmau [5]).

In literature, there are combinatorial characterizations of tree-like resolution size (by Beyersdorff, Galesi and Lauria [10]), resolution width (by Atserias and Dalmau [5]) and tree-like resolution space (by Impagliazzo and Pudlák [20]). We review the results from [5] (resolution width) and [20] (tree-like resolution space).

At last we show tradeoffs between various complexity measures (i.e, size, width and space) for tree-like resolutions (by Ben-Sasson [7]).

Contents

Chapter 1

Introduction

A language $L \subseteq \{0,1\}^*$ is in complexity class P iff the function $f_L : \{0,1\}^* \to \{0,1\}$ is computable in polynomial time, where $f_L(x) = 1 \iff x \in L$. A language L is in complexity class NP iff all strings in L have a short, polynomial time checkable proof of membership in L [1]. Intuitively the class P contains decision problems that can be efficiently solved and class NP contains decision problems whose solutions can be efficiently verified. It is still open whether or not the two classes are the same. Cook in the 1970's suggested that proof complexity bounds would give us answer for this problem. In particular he along with Reckhow proved in [2] that if one can find a polynomial-size family of tautologies that does not have polynomial size proofs then this will separate NP from $coNP$ [1](i.e, this will show that NP is not closed under complement) and thus separate P from NP (as P is closed under complement). Since finding such a family of tautologies is quite hard, the theory of proof complexity breaks this problem into smaller problems of proving such lower bounds for specific proof systems. Solving these smaller problems has useful implications for automated theorem proving as well.

1.1 Proof Systems

Consider the language SAT of all satisfiable boolean formulas. Trivially SAT is in NP as there exists always a short proof (satisfying truth assignment) for formulas in SAT. However how short proofs for formulas not in SAT could be is not clear. This requires the definition of a proof system.

Definition 1.1. *[2] A proof system for a non empty language $L \subseteq \{0,1\}^*$ is a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $Rng(f) = L$. For string $x \in L$, we say a string $w \in \{0,1\}^*$ is f-proof of x if $f(w) = x$. We say a proof system for L is polynomially bounded if there exists a polynomial $p(x) \in \mathbf{N}[x]$ such that each $x \in L$ has an f-proof 'w' of size $|w| \leq p(|x|)$.*

With this terminology it is clear that NP is precisely the set of languages that have polynomially bounded proof systems. Define the language $TAUT$ to be the set of all propositional logic tautologies (i.e formula that evaluates to True on every assignment).

[1] For the formal definitions of complexity classes P, NP and $coNP$ see Appendix A.

Definition 1.2. *[2] A proof system for the language $TAUT$ is called a* **propositional proof system** *(pps).*

Definition 1.3 (Completeness and Soundness of a *pps*)**.** *We say that a pps 'Q' is complete if for each formula $\tau \in TAUT$ there exists a proof for τ in Q and is sound if the existence of a proof for τ in Q implies that $\tau \in TAUT$.*

Since a formula is unsatisfiable (i.e, evaluates to False on every assignment) iff its negation is a tautology, we can give the following equivalent definition of propositional proof systems. Define the language $UNSAT$ to be the set of all unsatisfiable propositional logic formulas.

Definition 1.4. *[2] A proof system for the language $UNSAT$ is called* **propositional proof system** *(pps).*

The following theorem is proved by Cook and Reckhow in [2]. We are presenting the proof following the lecture notes by Jan Krajiček [18].

Theorem 1.5. *[2] There exists a polynomially bounded pps iff $NP = coNP$*

Proof. Suppose polynomially bounded *pps* exist. Then there is a polynomially computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $Rng(f) = TAUT$ and there exists a polynomial $p(x)$ such that,

$$\forall \tau \in TAUT, \exists w(|w| \leq p(|\tau|)); f(w) = \tau$$

which is an NP definition of $TAUT$. Hence $NP = coNP$ as $TAUT$ is $coNP$-complete set.
On the other hand, if $NP = coNP$, then $TAUT$ is in NP and hence we have an NP definition for $TAUT$, which is, $\forall \tau \in TAUT, \exists u(|u| \leq q(|\tau|)); M(u,\tau) = 1$, where M is polynomially bounded TM (may be view as polynomially time function) and q is some polynomial. Using this fact one can define a polynomially bounded *pps* as follows:

$$f(w) = \begin{cases} \tau & \text{if } w = (u,\tau) \text{and } |u| \leq q(|\tau|) \wedge M(u,\tau) = 1 \\ 1 & \text{otherwise} \end{cases}$$

Here the domain of f is $\{0,1\}^*$. Interpret w as pair (u,τ), and if u is a proof of τ then f maps w to τ else f maps it to 1 which is trivially in $TAUT$. ■

Some examples of *pps* are 1) **Truth Tables** 2) **Resolution** 3) **Davis Putnam (DLL) Procedure** 4) **Frege Proofs** etc.
Truth table *pps* is just a naive way of proving that a given formula τ is in $TAUT$. In this system one evaluates τ on every possible assignments and produce a proof in terms of completely filled truth table. Checking correctness of the truth table (proof) is easy with respect to the size of the truth table, however the proof size itself is exponential which makes this system practically useless. That is, a truth table *pps* is not polynomially bounded.

1.2 Organization of the Report

The goal of this report is to understand resolution proof systems. The report is organized as follows:

- In Chapter 2, we formally define the resolution propositional proof system. We show the completeness and soundness of resolution proof system. We introduce tree-like resolution *pps* and show that it is complete and sound. We introduce the first complexity measure (size) for resolution and tree-like resolution *pps*.

- In Chapter 3, we show an exponential size lower bound for resolution. In particular, we show that proving pigeonhole principle is hard in resolution proof system. This result was first proved by Haken in 1985 [16]. We introduce the second complexity measure (width) for resolution and tree-like resolution proof systems, and show width-size relationship in resolution. The relation between width and size in resolution was proved by Ben-Sasson and Wigderson in 1999 [9]. At last we show that there exist formulas which have polynomial size resolution proofs but require exponential size in tree-like resolution sytem. This result was proved by Bonet and Galesi in 1999 [11].

- In Chapter 4, we introduce the third complexity measure (space) of resolution and tree-like resolution proof system and show its connection to both size and width. In particular, we show size-space relationship in tree-like resolution proof system. This result was proved by Esteban and Torán in 2001 [12]. We also show space-width relationship in resolution, proved by Atserias and Dalmau in 2003 [5].

  Several game-theoretic methods have been developed to understand the complexity of resolution proof system. We show a combinatorial characterization of resolution width, defined by Atserias and Dalmau [5]. We also show a combinatorial characterization of tree-like resolution space, defined by Impagliazzo and Pudlák [20].

- In Chapter 5, we show tradeoffs between various complexity measures (i.e, size, width and space) in resolution proof system. In particular, we show for tree-like resolution, there are formulas in which optimizing two of the measures (width-space or width-size) simultaneously is not possible. This result was proved by Ben-Sasson [7].

- We conclude the report in Chapter 6.

- Appendix contains all the useful notations used in the report.

Chapter 2

Resolution

Resolution propositional proof system (*pps*) is the simplest *pps* for which it is not easy to prove a lower bound. This system was first introduced by Blake in 1937 [1] but Davis along with Putnam in 1960 [19] and Robinson in 1965 [3] were the first to use it in automated theorem proving.

2.1 Definitions

We will consider formulas over a set of propositional variables V. A literal is either a variable $x \in V$ or its negation $\neg x$. A term is a conjunction $l_1 \wedge \cdots \wedge l_k$ of literals (WLOG, distinct literals) and a formula F in disjunctive normal form (DNF) is a disjunction ($\vee$) of terms. A clause C is a disjunction of literals (WLOG, distinct literals) $l_1 \vee \cdots \vee l_k$ and a formula F in conjunctive normal form (CNF) is a conjunction ($\wedge$) of clauses C_i's. For convenience the clause C is written simply as a set of literals $\{l_1, \ldots, l_k\}$ and any CNF formula as a set of clauses.

Resolution is a proof system for proving that boolean formulas in a DNF form are tautologies. We know that transforming a boolean formula into an equivalent one in the DNF form may increase its size exponentially. However, we do not need an equivalent formula, we only need that the original formula is a tautology iff the constructed DNF formula is too. One way to achieve this is as follows: given a general formula ϕ, transform $\neg\phi$ into a CNF formula $\neg\phi_C$ using polynomial reduction which preserves satisfiability. The complement of the CNF formula $\neg\phi_C$ is the required DNF formula, since $\phi \in TAUT \iff \neg\phi \in UNSAT \iff \neg\phi_C \in UNSAT \iff \neg(\neg\phi_C) \in TAUT$. Hence we can assume that the given formula is in DNF form.

In order to prove that the given DNF formula A is in $TAUT$, resolution solves the equivalent complement problem of proving that the CNF formula $\neg A$ is in $UNSAT$. Thus resolution is a refutation proof system in CNF formulas. The only inference rule in this proof system is the resolution rule:

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

The variable x is called the resolved variable and we say that x has been resolved from the hypothesis clauses $C \vee x$ and $D \vee \neg x$ to get the resolvent (conclusion) clause $C \vee D$. We say a clause is satisfied by an assignment if it makes true at

least one literal in the clause. We will denote the empty clause by $\square$. The empty clause can not be satisfied by any assignment. It is clear that if both the clauses in the hypothesis of resolution rule are satisfied by an assignment then the assignment satisfies the conclusion too. Hence the resolution rule is sound.

Let $F = T_1 \vee \cdots \vee T_k$ be a DNF formula in $TAUT$, where $T_i = l_1^i \wedge \cdots \wedge l_{n_i}^i$ and n_i is the number of literals in term T_i for each $i \in I = [1..k]$. Define clauses $C_i = \{\neg l_j^i | 1 \leq j \leq n_i\}$ for each $i \in I$. Clearly the CNF formula $\{C_1, \ldots, C_k\}$ is the complement of F and is in $UNSAT$ iff F is in $TAUT$. A resolution proof of F is a sequence of clauses $D_1, \ldots, D_t$ such that:

1. Each $D_q, 1 \leq q \leq t$ is either one of initial clauses $C_i, i \in I$, or is derived from some clauses D_m, D_n with $m, n < q$ using the resolution rule.

2. The last clause (i.e, D_t) is the empty clause ($\square$).

The existence of such a proof certifies that the clauses C_i's can not be simultaneously satisfied. Hence often called as a refutation of $\{C_1, \ldots, C_k\}$.

In any resolution refutation $\pi = D_1, \ldots, D_t$, of a DNF formula $F \in TAUT$, if we store pointers from each D_q to D_m, D_n such that $m, n < q$ and D_q is a conclusion of a resolution rule with hypothesis D_m, D_n, then we actually get a directed acyclic graph G_π. In G_π, the vertices are the clauses and the edges are from the resolvent (conclusion) of an inference to the two hypothesis (the orientation may be defined in the reverse order, i.e, from the hypothesis to the conclusion). We call G_π as a proof graph of F. The resolution proof π of F is a sequence of vertices of G_π in a topological order.

If the last clause $D_t = A \neq \square$, then we say that π is a derivation of clause A from a CNF formula $\neg F$. It shows that every assignment satisfying $\neg F$ also satisfies A.

2.2 Completeness and Soundness of Resolution

The next theorem states that resolution proof system is sound and complete. We are presenting the proof from the lecture notes by Jan Krajiček [18].

Theorem 2.1. *A DNF formula has a proof in resolution iff it is a tautology.*

Proof.
Soundness Let F with k terms be a DNF formula that has a proof in resolution. We will show that F is a tautology by showing that $\neg F$ is unsatisfiable. Let C_i's be the clauses obtained as above. Any truth assignment satisfying all C_i's, by the soundness of resolution rule, would have to satisfy all the clauses in any resolution refutation of C_i's, in particular, the end (empty) clause as well. But that is not possible as there are no literals in $\square$.

Completeness Let F a DNF formula with k terms is a tautology. Hence $\neg F$, in particular, the set $\mathcal{C} = \{C_1, \ldots, C_k\}$ is unsatisfiable. Let $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$ be the literals appearing in $\mathcal{C}$. We will prove by induction on n that for any such $\mathcal{C}$ there is a resolution refutation of $\mathcal{C}$.

When $n = 1$, $\mathcal{C}$ must contain clauses (x_1) and $(\neg x_1)$ and their resolvent is the empty set. Assume $n > 1$, and partition set $\mathcal{C}$ of clauses into four disjoint subsets:

$\mathcal{C}_{00}, \mathcal{C}_{01}, \mathcal{C}_{10}, \mathcal{C}_{11}$, of those clauses which contain no x_n and no $\neg x_n$, no x_n but do contain $\neg x_n$, do contain x_n but not $\neg x_n$ and contains both x_n, $\neg x_n$ respectively. Observe that the set $\mathcal{C}$ is unsatisfiable even without the fourth subset $\mathcal{C}_{11}$ of clauses, since every assignment satisfies all the clauses in $\mathcal{C}_{11}$. Thus fourth subset is redundant for unsatisfiability. Now using the first three disjoint subsets and the resolution rule we will construct a new set $\mathcal{C}'$ of clauses such that it does not contain both x_n and $\neg x_n$ and is also unsatifiable. Thereafter, by invoking induction hypothesis we are through.

Construct the new set of clauses $\mathcal{C}'$ as follows:

1. Include set $\mathcal{C}_{00}$ in $\mathcal{C}'$ (i.e, include all the clause from the set $\mathcal{C}_{00}$)

2. Include in $\mathcal{C}'$ all clauses $C_1 \vee C_2$ that are obtained by the resolution rule applied to pairs of clauses $C_1 \vee \neg x_n$ from $\mathcal{C}_{01}$ and to $C_2 \vee x_n$ from $\mathcal{C}_{10}$.

Note that the clauses from subset $\mathcal{C}_{00}$ and the new clauses introduced in the 2nd step of the construction contains neither x_n nor $\neg x_n$. Moreover, the new set of clauses $\mathcal{C}'$ is also unsatisfiable: if not, then there is an assignment $\alpha' : \{x_1, \ldots, x_{n-1}\} \to \{0, 1\}$ which satisfies all the clauses in $\mathcal{C}'$. Clearly α' satisfies either I) all the clauses C_1 such that $C_1 \vee \neg x_n \in \mathcal{C}_{01}$ or II) all clauses C_2 such that $C_2 \vee x_n \in \mathcal{C}_{10}$ (otherwise we could find some $C_1 \vee C_2 \in \mathcal{C}'$ not satisfied by α'). Using this fact one can easily extend α' to a truth assignment α satisfying $\mathcal{C}$, which is a contradiction. The extension is simple, for the case I assign value 1 to x_n and for case II assign value 0 to x_n. ■

2.3 Tree-like Resolution

Let F be an unsatisfiable CNF formula. A resolution proof $\pi = (D_1, \ldots, D_t)$ of F is tree-like iff each D_i is used at most once as a hypothesis of an inference in the proof. If one draws the proof graph of π, a directed graph with vertices being the clauses and the edges going from the conclusion of an inference to the two hypothesis, then the condition tree-like precisely says that the graph is a tree (a proof-tree).

The proof system allowing exactly tree-like resolution proofs is called tree like resolution proof system. Several other restrictions of the resolution proof system have appeared in the literature [21]. Hereby, the term '**general** resolution proof system' will be used to denote resolution systems without any additional restriction.

It can be easily seen that tree-like resolution proof system is also sound and complete (since from general resolution refutation one can get tree-like refutation just by rederiving each time the clauses that are needed more than once in the refutation from the initial clauses).

2.4 Resolution Proofs with Weakening Rule

In resolution proof system some times we also include one more inference rule, the weakening rule:

$$\frac{C}{C \vee D}$$

where C and D are arbitrary clauses. To be precise, let $F = T_1 \vee \cdots \vee T_k$ be a DNF formula in $TAUT$ and the CNF formula $\{C_1, \ldots, C_k\}$ be the complement of F as before. Clearly the CNF formula is unsatisfiable iff F is a tautology. A resolution proof with weakening rule of F is a sequence of clauses $D_1, \ldots, D_t$ such that the last clause is the empty clause ($\square$) and each clause D_q is either some initial clause $C_i, i \in [k]$ or is derived from the previous clauses using one of the following inference rules:

1. The resolution rule: $\frac{D_m \quad D_n}{D_q}$, where $m, n < q$, D_m, D_n are clauses and x is a variable such that $x \in D_m$ and $\neg x \in D_n$ and $D_q = D_m \cup D_n - \{x, \neg x\}$.

2. The weakening rule: $\frac{D_m}{D_q}$, where $m < q$, D_m is one of the previous clause and $D_q = D_m \vee C$ for an an arbitrary clause C.

Remark 2.2. *A resolution proof system with weakening rule is sound and complete: the weakening rule $\frac{C}{C \vee D}$ is sound since any assignment which satisfies C must also satisy $C \vee D$. On the other hand from Theorem 2.1 we know that resolution proof system with out weakening rule is complete which implies that resolution proof system with weakening rule is also complete.*

For any resolution proof $\pi = \{D_1, \ldots, D_t\}$ of $F \in TAUT$ with weakening rule we have a directed acyclic graph G_π where vertices corresponds to clauses and edges are from the conclusion to the hypothesis of an inference rule. Note that in G_π the indegree of each vertex can be either $0, 1$ or 2. Analogous to the definition of tree-like resolution proof we have tree-like resolution proof with weakening rule.

The following lemma shows how to remove weakening rule from a tree-like resolution proof with weakening rule.

Lemma 2.3. *Let F be an unsatisfiable CNF formula and let π be a tree-like refutation of F with weakening rule. Then we can convert π to π' such that π' is a tree-like resolution refutation of F without weakening rule.*

Proof. Let T_π be the corresponding tree graph for π. The root of T_π is the empty clause. Assume the edges of T_π are oriented towards the root. If none of the clauses in π uses weakening rule then π is our π' and we are done. Otherwise there is a clause in π which uses weakening rule. Consider a highest clause in T_π which is derived via weakening rule. Let this be the clause $E = C \vee D$ with its unique child C. Consider the unique path ρ from E to the root of T. As E is the highest clause it is guarenteed that no clause in ρ is derived using weakening rule. Since the root of T has no literals, there must be clauses in ρ where literals of C and D are chopped off one by one using resolution rule. Call those clauses as C' and D' respectively. While chopping off literals from C and D new literals may also gets added. In ρ these new literals also gets removed subsequently by some clauses, Call them C'_N and D'_N respectively.

Consider the tree T constructed from T_π as follows: from the path $C\rho$ in T_π remove the clause E along with all the clauses of type D' and D'_N. Concatenate the remaining clauses of $C\rho$ to get $C\rho'$. Each of the clauses D' and D'_N are derived using resolution rules such that one of their hypothesis clauses are from ρ and the other-one from outside ρ. Remove the outer clause as well along with its entire

subtree. Since these subtrees are used only to resolve the literals of D and the new literals, these can be removed safely. Keep the remaining tree T_π as it is to get T. We claim that T is a valid tree-like refutation of F which uses one less weakening rule: as we have removed the clause E, the weakening rule used to derived it from C has been removed. Moreover now the literals in D are not there in ρ hence we can remove safely all the clauses used to remove it (i.e, D') and D'_N.

By repeating this procedure in top down order we can construct a tree-like refutation T' of F without using the weakening rule. ■

In this report we always assume that resolution proofs uses only the resolution rule to derive the empty clause, unless stated otherwise.

2.5 Complexity Measure

The most important complexity measure of a resolution proof is its minimal size, measured in the number of clauses appearing in a proof. More formally, let F be an unsatisfiable CNF formula. Let $\pi \vdash F$ (resp. $\pi \vdash_{tl} F$) denote that π is a general (resp. tree-like) resolution refutation of F. The size $|\pi|$ of a refutation π in any of the above two systems is defined as the number of clauses used in π. The size complexity $S(\vdash F)$ (resp. $S_T(\vdash F)$) of deriving a CNF formula F in general resolution (resp. in tree-like resolution) is defined as $\min_{\pi \vdash F} |\pi|$ (repectively $\min_{\pi \vdash_{tl} F} |\pi|$). It is clear that for any unsatisfiable CNF F we have $S(\vdash F) \leq S_T(\vdash F)$. The problem of proving a size lower bound for general resolution is to come up with an unsatisfiable CNF formula F such that $S(\vdash F)$ is at least superpolynomial or exponential. This problem has recieved much attention over the past few decades and various size lower bounds have been obtained for various families of formulas in CNF form. We will go through some of these results in the next chapter. We end this chapter with an upper bound on $S(\vdash F)$.

Simple analysis of Theorem 2.1 shows the following:

Lemma 2.4. *Let F be an unsatisfiable CNF formula over n variables then there exists a resolution refutation π of F such that depth of $G_\pi \leq n$.*

Proof. Resolution refutation π constructed in Theorem 2.1 resolves at least one variable in each step hence there are at most n levels in the proof graph G_π. One can easily convert G_π into a tree T without increasing its depth. Hence there also exists a tree-like refutation T of F such that depth of T is $\leq n$. ■

Lemma 2.5. *For every unsatisfiable CNF formula F on n variables, $S(\vdash F) \leq 2^{n+1}$.*

Proof. There are n variables, so $2n$ literals, so 2^{2n} possible clauses. So the refutation can not be larger than 2^{2n}; this is a trivial upper bound on $S(\vdash F)$.

However, the way we construct the refutation in the proof of Theorem 2.1, we eliminate a variable at each step, getting a depth n graph. Even if we were to write it as a tree, a binary tree of depth n has at most $1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$ vertices. So the refutation is actually of size at most 2^{n+1}. ■

Chapter 3

Lower Bounds for Resolution

The first step in proving a lower bound for any proof system is often to find a candidate hard example for that proof system. In case of general resolution proof system the first superpolynomial (and, in fact, exponential) size lower bound has been proved in 1985 by Haken [16]. The tautology he considered in his proof is elementary yet basic to mathematics: the pigeonhole principle.

3.1 The Pigeonhole Principle

The pigeonhole principle says that if we put m pigeons into n holes, where $m > n$, then at least one hole must contain more than one pigeon. Mathematically, it asserts that if $m > n$ then there is no one-to-one mapping from a set of size m to a set of size n.

3.1.1 Encoding the Pigeonhole Principle as an Unsatisfiable CNF Formula

This can be easily encoded as an unsatisfiable CNF formula PHP_n^m over variables $P_{i,j}$, $1 \leq i \leq m, 1 \leq j \leq n$, which is supposed to be assigned "True" if pigeon i is put into hole j. PHP_n^m contains the following clauses:

(1) $(P_{i,1} \vee P_{i,2} \vee \ldots, P_{i,n})$, for $1 \leq i \leq m$. This clause ensures that i^{th} pigeon is assigned to some hole.

(2) $(\neg P_{i,j} \vee \neg P_{k,j})$, for $1 \leq i < k \leq m$, $1 \leq j \leq n$. This clause ensures that the j^{th} hole does not get both the i^{th} and the k^{th} pigeon.

(3) $(\neg P_{i,j} \vee \neg P_{i,k})$, for $1 \leq i \leq m$, $1 \leq j < k \leq n$. This clause ensures that the i^{th} pigeon should not be assigned to both the j^{th} and the k^{th} hole.

Thus there are total $m + n\binom{m}{2} + m\binom{n}{2}$ clauses (which is less than $3m^2$ when $m > n$). The clauses collectively ensure that any satisfying assignment to theses variables corresponds to a valid one-to-one function from m pigeons to n holes. Thus if $m > n$ then due to the pigeonhole principle the CNF formula PHP_n^m is unsatisfiable.

3.1.2 Inductive Proof of the Pigeonhole Principle

Proof. In order to prove pigeonhole principle for m pigeons and n holes where $m > n$, it is sufficient to prove it for any n pigeons and $n-1$ holes. Because then we can arbitrarily choose $n+1$ pigeons from m pigeons and show the nonexistence of any one-to-one function $f : n+1 \to n$, which impies the nonexistence of any one-to-one function for m pigeons to n holes as well.

We will prove this by induction on n. For the base case, it is trivial that there is no one-to-one function from 2 pigeons to 1 hole. For the induction step, pick any function $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n-1\}$ and we will show that f is not one-to-one function. For this, pick any hole y. If f has mapped more than one pigeons in hole y, then we are through, i.e, f is not a one-to-one function. If f has mapped exactly one pigeon say x to the hole y (i.e, for unique $x : f(x) = y$), then we throw away the pigeon x and the hole y. Now we are left with only $n-1$ pigeons and $n-2$ holes and from induction hypothesis we are done. The final case is when hole y is empty, that is, f has mapped none of the pigeons to hole y. In this case we throw away the hole y. Now we are left with only $n-2$ holes. By induction hypothesis, we know that there is no one-to-one function from $n-1$ pigeons and $n-2$ holes, which again implies that there can not be any one-to-one function from n pigeons to $n-2$ holes. ■

3.1.3 Lower Bound for General Resolution: PHP^n_{n-1}

We will show that general resolution requires exponential size to prove the pigeonhole principle. This was the first exponential lower bound shown for general resolution.

Theorem 3.1. *For any $n \geq 2$, every resolution refutation of PHP^n_{n-1} has size at least $2^{n/20}$. That is $S(\vdash PHP^n_{n-1}) \geq 2^{n/20}$.*

The correctness of any resolution refutation can be tested by assigning values to its variables. A correct refutation shows that no assignment can satisfy all the initial set of clauses. To be precise, given a refutation π and an assignment α, assign values to its variables based on α. Evaluate the vertices (clauses) of π based on α to get a label 0 or 1. Call a path consiting of 0 label clauses from the root (empty) clause to some leaf (initial) clause of π a 0-path. We know a refutation is a derivation of an empty clause from the set of initial clauses using only resolution rules, and since resolution rule is sound, if we assume that the initial clauses are true then the empty clause must be true. As no assignment can satisfy the empty clause, for every assignment there must be a 0-path from the empty clause to one of the source vertex in the refutation. The 0-path shows that the given assignment does not satisfy the corresponding initial clause.

Instead of producing a 0-path for all possible assignments consider refutations that only produces a 0-path for certain subset of assignments, that is, which shows that a certain subset of assignments can not satisfy all the initial clauses simultaneously. Of course for assignments outside this subset the refutation may or may not produces a 0-path. Thus this is a relaxation in the notion of resolution refutation. Hence we call such a refutation a **relaxed refutation**. Observe that any lower bound on the size of relaxed refutation also apply for the general refutation.

The subset of assignments for which the relaxed refutation produces a 0-path are said to be critical assignments.

Critical assignment for PHP^n_{n-1} An assignment is said to be critical if it corresponds to a mapping that map $n-1$ pigeons to $n-1$ holes in a one-to-one manner and leaves the n^{th} pigeon unassigned. There are $n!$ such assignments. If the index of the only unassigned pigeon is k we call such a critical assignment a k-critical assignment. We also call critical assignments as **test assignments**.

Any relaxed refutation derives contradiction only for critical assignments. This relaxation allows us to make all clauses in the refutation monotone (i.e, with no occurence of negated variables). For this we will perform the following transformation: For each clause C in the refutation, produce a monotonized clause $M(C)$ by replacing each negated variable $\neg P_{i,j}$ by $\bigvee_{l \neq i} P_{l,j}$. We call the resulting refutation as **monotonized resolution refutation** (MRR).

Claim 3.2. *$M(C)$ is satisfied by exactly the same set of test assignments as the original clause C.*

Proof. First we will show that any critical assignment α which satisfies C must also satisfy $M(C)$: as the transformation does not change any positive literals, if α satisfied C by setting some positive literal to 1 then we are done. Now suppose α satisfied C by assigning $\neg P_{i,j} = 1$. Thus $P_{i,j} = 0$ which implies that i^{th} pigeon is not in j^{th} hole. But α is critical assignment, therefore $\exists k$, such that k^{th} pigeon is in j^{th} hole. Thereby $P_{k,j} = 1$. Since $P_{k,j}$ is in $M(C)$, $M(C)$ is satisfied by α.

On the other hand, suppose a critical assignment α falsifies C. We will show that α also falsifies $M(C)$: again we only need to consider negative literals of C. Suppose α assigns $\neg P_{i,j} = 0$. Thus $P_{i,j} = 1$ which implies i^{th} pigeon is in the j^{th} hole. Since α is critical assignment, none of the other pigeons are assigned to hole j by α, there by $\forall k, k \neq i$, $P_{k,j} = 0$. So $M(C)$ is also falsified by α.

■

For any critical assignment α, a relaxed resolution refutation produces a 0-path (i.e, path of unsatisfiable clauses from the empty clause to some source clause). And due to claim 3.2, the corresponding monotonized resolution refutation also preserves the 0-path from the empty clause to the source clause for α. Thus a monotonized resolution refutation also shows via a 0-path that any critical assignment does not satisfies all the initial clauses simultaneously. The next Lemma shows that monotonized refutations must have a large clause.

Lemma 3.3. *Every monotonized resolution refutation (MRR) π of PHP^n_{n-1} must contains a clause with at least $2n^2/9$ variables.*

Proof. For each clause in the monotonized refutation, let

$$witness(C) = \{i : \text{ there is an } i\text{-critical assignment } \alpha \text{ falsifying } C\}$$

Define the complexity of a clause C, $comp(C)$ to be $|witness(C)|$. Clearly $comp(C) \leq 1$ for $C \in PHP^n_{n-1}$. This is because the initial clauses of type (2) and (3) is satisfied by every critical assignment and the initial clauses of type (1) for

pigeon i is falsified by only i-critical assignments. Also $comp(\Box) = n$ since empty clause is falsified by any critical assignment. Let a clause C is derived from two clauses A and B (i.e, $\frac{A \quad B}{C}$), then $comp(C) \leq comp(A) + comp(B)$: since every critical assignment that falsifies C must falsify at least one of A and B. So $witness(C) \subseteq witness(A) \cup witness(B)$. Therefore $comp(C) \leq comp(A) + comp(B)$ (this shows that $comp$ is subadditive with respect to the resolution rule. Due to the above mentioned properties of $comp$, for every MRR the following claim holds:

Claim 3.4. *For every MRR π of PHP^n_{n-1}, there exists a clause C' such that $n/3 \leq comp(C') \leq 2n/3$.*

Proof. Consider the proof graph G_π of MRR π (edges are from the conclusion to the two hypothesis). The root of G_π corresponds to the empty clause and leaves corresponds to the source clauses of PHP^n_{n-1}. One can find C' with the following simple search algorithm in G_π: starting with the root vertex of G_π, go on descending towards source vertices by following the children with larger $comp$ values, until we encouter for the first time a clause A with $comp(A) < n/3$. The parent of A is C': as A is the first clause with smaller $comp$ values, we have $comp(C') \geq n/3$. Also among the two children say A and B we have, $comp(A) \geq comp(B)$. Thus $comp(B) < n/3$, and thereby $comp(C') \leq n/3 + n/3 = 2n/3$. ■

It is sufficient to show that for any clause C, if $comp(C) = p$ then C contains at least $p(n-p)$ distinct literals. Because then we have $comp(C') \in [n/3, 2n/3]$ and thereby C' must have at least $n/3(n - n/3) = 2n^2/9$ literals.

Fix any $i \in witness(C)$ and any i-critical assignment α which falsifies C. By definition, in α only pigeon i is not mapped to any hole. Fix any $j \notin witness(C)$. Construct a j-critical assignment α' from α just by swapping i by j. That is, in α, pigeon j is mapped to some hole say l, then in α' map pigeon i to hole l and leave only pigeon j unmapped. As $j \notin witness(C)$, the j-critical assignment α' must satisfy the clause C. Observe that the only difference between α and α' is that the value of $P_{i,l}, P_{j,l}$ are $0, 1$ respectively in α and $1, 0$ in α'. And since C is falsified by α but is satisfied by α', it must be the case that the variable $P_{i,l}$ must belongs to C.

Thus for the same i-critical assignment α and running over all $n - p$ values of $j \notin witness(C)$, we conclude that C contains at least $n - p$ distinct variables of the type $P_{i,l}$. As $comp(C) = p$, repeating the argument for every $i \in comp(C)$, we conclude that C contains at least $p(n-p)$ distinct variables. ■

Now we are ready to prove Theorem 3.1.

Proof of Theorem 3.1.
We are given any relaxed refutation π of PHP^n_{n-1}. Using above transformation convert π into a monotonized resolution refutation π'. As seen before π' correctly produces 0-path for every critical assignments. Observe that the transformation does not change the number of clauses in π. Thus $|\pi| = |\pi'|$. We need to show that $|\pi'| \geq 2^{n/20}$. Let us call a clause in π' large if it has at least $n^2/10$ variables. Let L be the number of large clauses in π'. We will show by contradiction that $L \geq 2^{n/20}$. This will prove the desired result.

Suppose not. Then we have $L < 2^{n/20}$. We define partial restrictions to variables that greatly reduces the number of large clauses. Since totally PHP^n_{n-1} has $n(n-1)$

variables and large clauses has at least $n^2/10$ variables. Thus there exists a variable $P_{i,j}$ which occurs in at least $1/10^{th}$ of the large clauses. (To verify this consider a matrix A whose rows corresponds to large clauses and columns to variables. Thus A has L rows and $n(n-1)$ columns. Define $A[i][j] = 1$ iff j^{th} variable belongs to the i^{th} large clause. Clearly each row of A has at least $n^2/10$ 1's. Thus total number of 1's row-wise in A is at least $(n^2/10) \cdot L$. Let each column of A has at most M 1's. Then clearly we have $n(n-1) \cdot M \geq (n^2/10) \cdot L$. Hence

$$\begin{aligned} M &\geq \frac{n^2 \cdot L}{10 \cdot n(n-1)} \\ &> \big(\frac{1}{10}\big) \cdot L, \text{ since } n \geq 2 \text{ we have } \frac{n}{n-1} > 1. \end{aligned}$$

This verifies the claim).

Define the following restriction: $P_{i,j} = 1$ and $\forall j' \neq j, P_{i,j'} = 0$ and $\forall i' \neq i, P_{i',j} = 0$. This restriction assigns pigeon i to hole j. It also ensures that pigeon i is not assigned to any other hole and no pigeon i' other than i is assigned to hole j. This restriction removed pigeon i and hole j from the contention. Hence we are left with only $n-1$ pigeons and $n-2$ holes. Also now we are left with only $n-1$ distinct j-critical assignments for $j \neq i$.

This restriction also sets all monotonized clauses in π' containing $P_{i,j}$ to true. Remove such clauses from π' along with all the variables $P_{i,j'}$ and $P_{i',j}$ from their corresponding clauses to get π_1. Clearly in π_1 at most $(9/10)L$ large clauses left. The claim is that π_1 is a monotonized resolution refutation for PHP^{n-1}_{n-2} (i.e, π_1 correctly produces a 0-path for all remaining critical assignments): after this restriction we are left with only $n-1$ pigeons and $n-2$ holes. Now there is no i-critical assignments as pigeon i is already assigned to hole j. In other words all the clauses corresponding to pigeon i and hole j already gets satisfied by the restriction. And from π' we have only removed the clauses which becomes 1 after the restriction along with variables which were mentioning about the i^{th} pigeon and the j^{th} hole to get π_1. Thus π_1 still preserves the 0-path from the empty clause to some source clause for each j-critical assignment with $j \neq i$. Hence π_1 correctly produces a 0-path for all critical assignment for the formula PHP^{n-1}_{n-2}.

Similarly repeat the procedure t many times to get an MRR π_t of PHP^{n-t}_{n-1-t} with at most $\big(\frac{9}{10}\big)^t L$ large clauses.We pick t such that $\big(\frac{9}{10}\big)^t L < 1$. Simple calculation gives us such a t:

$$\begin{aligned} \big(\frac{9}{10}\big)^t L &< 1 \\ L &< \big(\frac{10}{9}\big)^t \\ \log_2 L &< t \log_2(10/9) \\ \frac{\log_2 L}{\log_2(10/9)} &< t \end{aligned}$$

By assumption we have $L < 2^{n/20}$. Thus we pick

$$t > \frac{\log_2 2^{n/20}}{\log_2(10/9)} = \frac{n}{20 \cdot \log_2(10/9)} > \frac{\log_2 L}{\log_2(10/9)}$$

and repeat the procedure t many times to get an MRR π_t of PHP^{n-t}_{n-1-t} with no large clauses. Therefore the largest clause in π_t has strictly less than $n^2/10$ variables. Also from Lemma 3.3, π_t must have a clause with at least $2(n-t)^2/9$ variables. Combining the above two statements we have,

$$\frac{2(n-t)^2}{9} \leq \text{number of variables in largest clause of } \pi_t \ < \frac{n^2}{10} \tag{3.1}$$

Thereby we have,

$$\begin{aligned}
\frac{2(n-t)^2}{9} &< \frac{n^2}{10} \\
\frac{(n-t)^2}{n^2} &< \frac{9}{20} \\
\left(1 - \frac{t}{n}\right)^2 &< \frac{45}{100} \\
-\frac{t}{n} &< \frac{3\sqrt{5}}{10} - 1 \\
\therefore t &> n\left(1 - \frac{3\sqrt{5}}{10}\right)
\end{aligned}$$

Thus from 3.1 we also have

$$t > n\left(1 - \frac{3\sqrt{5}}{10}\right)$$

Now if we pick $t = n(0.3290)$ then this will contradict Equation 3.1 and hence Lemma 3.3. Therefore it must be the case that $L \geq 2^{n/20}$.

■

Remark 3.5. *The above proof forces us to choose* $t > \frac{n}{20 \cdot \log_2(10/9)} = n(0.32894)\Big($ *in order to keep the number of variables in largest clause of* $\pi_t < \frac{n^2}{10}$ $\Big)$ *and* $t > n\left(1 - \frac{3\sqrt{5}}{10}\right) = n(0.32917)\Big($ *in order to have at least one clause with* $\geq \frac{2(n-t)^2}{9}$ *variables* $\Big)$*. Thus we have a contradiction if we pick* $t = n(0.3290)$*. This is because* $t = n(0.3290)$ *satisfies the first condition but not the second as* $n(0.32894) < t = n(0.3290) < n(0.32917)$*.*

3.2 Width of a Resolution Proof

A second complexity measure of a resolution proof is the minimal width, measured as the maximal number of literals of a clause in the proof. More formally, let us define the width of a clause C to be the number of literals in C, denoted by $width(C)$. The width of a CNF formula F, denoted as $w(F)$, is defined to be the width of a largest clause in F, i.e, $w(F) = \max_{C \in F}\{width(C)\}$. The width $w(\pi)$ of a refutation π, is

defined as the width of a largest clause appearing in π, i.e, $w(\pi) = \max_{C \in \pi}\{width(C)\}$. The width $w(\vdash F)$ (resp. $w(\vdash_{tl} F)$) of deriving an unsatisfiable CNF formula F in general (resp. tree-like) resolution is defined as $\min_{\pi \vdash F}\{w(\pi)\}$ (resp. $\min_{\pi \vdash_{tl} F}\{w(\pi)\}$).

This measure was first introduced by Galil in 1977 [14]. In 1999, Ben-sasson and Wigderson, based on ideas of Clegg, Edmonds and Impagliazzo, relate size lower bounds of resolution proof to width lower bounds [9]. The main point of their results is that now, to prove size lower bounds, it is sufficient to prove width lower bounds.

3.2.1 Width and Tree-like Size

In this section we give a relation between $w(\vdash F)$ and $S_T(\vdash F)$ via Theorem 3.6. The Theorem was proved in [9]. Here we are presenting the proof from the survey, 'Proof Complexity' by Paul Beame [6].

Theorem 3.6. *[9] Let F be an unsatisfiable CNF formula and $S_T(\vdash F) = S$. Then $w(\vdash F) \leq (\lfloor \log_2 S \rfloor + w(F))$. In particular, any smallest size tree-like resolution proof π of F can be converted to another tree-like resolution proof π' of F, of width at most $(\lfloor \log_2 S \rfloor + w(F))$.*

Proof. For simplicity, let $w = (\lfloor \log_2 S \rfloor + w(F))$. We will show this by induction on the size of the smallest tree-like resolution proof π of F.

Base case When $S = 1$, we know that π has only one vertex, which has to be a vertex for the empty clause. Thus consider π' to be π itself and we have $w(\pi') = 0 \leq w = (\lfloor \log_2 1 \rfloor + w(F))$.

Induction hypothesis Assume that for all sets F'' of clauses with a smallest tree-like resolution refutation of size $S'' < S$, there is a tree-like resolution proof π'' of F'' with $w(\pi'') \leq \lfloor \log_2 S'' \rfloor + w(F'')$.

Induction step Now let π be the smallest size tree-like resolution proof of F with $|\pi| = S$. Consider the proof graph, which is a tree T, of π. Let x be the last variable resolved on to derive the empty clause $\Box$, and T_1, T_2 be the two subtrees from the root. As the number of vertices in T is S, sizes of both the subtrees on top can not be strictly more than $S/2$ at the same time, therefore one of the two subtrees at the top must has size at most $S/2$ and the other has size strictly smaller than S. Without loss of generality, let these be the left (T_1) and the right (T_2) subtree, respectively. Also assume that $\neg x$ comes from T_1 and x from T_2. See Figure 1(a).

Let $F|_{x=1}$ and $F|_{x=0}$ be the CNF formulas obtained after assigning 1 and 0 respectively to variable x in F. Clearly in $F|_{x=1}$ (resp. $F|_{x=0}$) all the clauses containing x (resp. $\neg x$) becomes 1 and $\neg x$ (resp. x) has been dropped from all the clauses containing it.

As T is the smallest size tree-like refutation, T_1 is a tree-like derivation of $\neg x$ from F in a smallest possible size ($\leq S/2$). And due to Lemma 3.8 (see below) we have a refutation (i.e, derivation of $\Box$) of $F|_{x=1}$ of size $\leq S/2$. Call the corresponding tree-graph as T_1'. By applying induction hypothesis on the refutation T_1', we have a

tree-like refutation (T_1'') of $F|_{x=1}$ with width $\leq \lfloor \log_2 S/2 \rfloor + w(F|_{x=1}) \leq \lfloor \log_2 S \rfloor + w(F) - 1 = w - 1$.

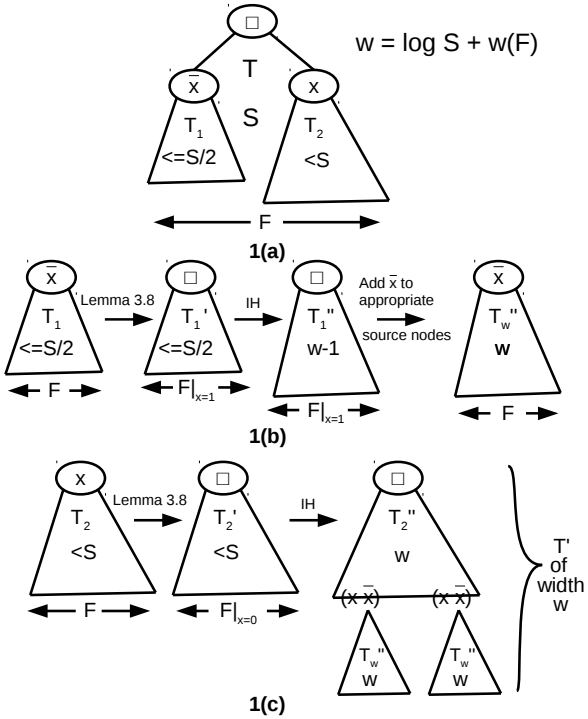

Figure 1: (a) Smallest tree-like proof T of F. Smallest tree-like derivation T_1 of $\bar{x}$ of size <= S/2. Simlilarly T_2 is a smallest derivation of x of size < S. (b) Using T_1, Lemma 3.8 and induction hypothesis construct a proof of F or derivation of $\bar{x}$ with width at most w. (c) If (b) fails to construct the proof, use T_2, Lemma 3, induction hypothesis and T_w" to construct a proof T' of width atmost w.

Let $F_{\neg x} \subseteq F$ be the set of all clauses containing $\neg x$. As T_1'' uses the initial clauses from $F|_{x=1}$, there is a subset of initial clauses (F') of $F_{\neg x}$ with $\neg x$ being removed used by T_1''. From the proof T_1'' constructs a derivation (T_w'') by adding back $\neg x$ in all the initial clauses F' and propogate them up by mimicking exactly the steps taken in T_1''. As there are no resolution steps in T_1'' which resolves on the variable x, T_w'' is a valid derivation. Moreover the width of T_w'' increases by one. That is $w(T_w'') = w(T_1'') + 1 \leq w$. There can be two cases with T_w'': either the empty clause or $\neg x$ is on the top. In the first case T_w'' is actually a refutation of F with $w(T_w'') \leq w$ and we are done. Otherwise we have a derivation of $\neg x$ from F with $w(T_w'') \leq w$. See Figure 1(b).

Repeat the above steps for the right subtree of T. As we have a smallest size tree-like derivation (T_2) of x from F of size $< S$, we have a tree-like refutation (T_2') of $F|_{x=0}$ of size $< S$ due to Lemma 3.8. Again by applying induction hypothesis to T_2' we have a refutation (T_2'') of $F|_{x=0}$ with width $\leq (\lfloor \log_2 S \rfloor + w(F_{x=0})) \leq w$. Let $F_x \subseteq F$ be the subset of all the clauses of F containing x and let F'' be the subset of F_x with x removed which are used as the initial clauses in T_2''. Now using T_2'' and T_w'' we will construct a tree-like refutation T' of F with $w(T') \leq w$ as follows: add x in all the clauses of F'' and resolve them before the leaf level of T_2'' by using the copy of the tree-like derivation T_w'' of $\neg x$. See Figure 1(c). By doing this we again get back

F'', which we can used by T_2'' to derive an empty clause. Since $w(T_2''), w(T_w'') \leq w$ it implies that $w(T') \leq w$.

■

Corollary 3.7. *Any tree-like resolution proof of an unsatisfiable CNF formula F requires size at least $2^{(w(\vdash F)-w(F))}$, i.e, $S_T(\vdash F) \geq 2^{(w(\vdash F)-w(F))}$.*

Proof. Let $S_T(\vdash F) = S$. From theorem 3.6, we have a tree-like refutation π' of F with width $w \leq (\lfloor \log_2 S \rfloor + w(F))$. Thus we have,

$$\begin{aligned} w(\vdash F) &\leq \lfloor \log_2 S \rfloor + w(F) \\ &\leq \log_2 S + w(F) \\ w(\vdash F) - w(F) &\leq \log_2 S \end{aligned}$$

Exponentiating both side we have the desired result,

$$S_T(\vdash F) = S \geq 2^{(w(\vdash F)-w(F))}.$$

■

Now we give the proof of Lemma 3.8 which will complete the proof of Theorem 3.6.

Lemma 3.8. *For an unsatisfiable CNF formula F, if we have a smallest size tree-like derivation of $\neg x$ from F of size S, then we have a refutation of $F|_{x=1}$ of size S.*

Proof. Let π be the smallest size tree-like derivation of $\neg x$ from F with $|\pi| = S$, and T be the corresponding tree proof. If x has not appeared in any of the clauses in π, then it is guarenteed that no resolution step is taken to resolve on the variable x. Thereby if we remove the literal $\neg x$ from all the clauses of T where it appears, then clearly after this modification at the top of T, we have a unique empty clause due to minimality. Moreover, the resulting tree T', with $|T'| = S$ is a tree-like refutation of $F|_{x=1}$. This is because by assumption the initial clauses in T' do not have any clauses from F containg x and now all its initial clauses do not contain $\neg x$ as well. Also all the resolution steps in T' exactly correspond to the resolution steps in T which are all known to be valid.

Thus it is sufficient to proof that variable x does not belongs to any of the clauses of T. We will prove this by contradiction. By assuming that x belongs to some clauses of T we will come up with a shorter derivation of $\neg x$ from F which is a contradiction.

Suppose not. Then x appears in some clauses of T. As x is not in the top clause of T, it must get resolved using the resolution rule on the variable x. Consider a highest clause in T where x gets resolved. Let that clause be $E = (D \vee C)$ and $P = D \vee \neg x, Q = C \vee x$ be its two children. Consider the unique path ρ from E to the root of T. Observe that the root contains only $\neg x$, and in E, there are no $x, \neg x$. This means there are clauses in ρ where literals of D and C are chopped off one by one. Call those clauses as D', C' respectively. While chopping off literals from D, new literals may gets added. In ρ these new literals also gets removed in some

clauses call them D'_N. Similarly we have C'_N. Note that in the process of removing literals from D and C for sure $\neg x$ gets added and x does not get added.

Consider the tree T' constructed from T as follows: from the path $P\rho$ in T remove the clause E, along with all the clauses of the type C' and C'_N. Concatenate all the remaining clauses of the path $P\rho$ to get a path $P\rho'$. Keep the remaining tree T as it is to get T'. Clearly $|T'| < |T|$. We claim that T' is a valid tree-like derivation of $\neg x$: since P contains $\neg x$, $\neg x$ remains in all the clauses in the path $P\rho'$ because E was the highest clause resolving on x and that has already been removed. This shows that $\neg x$ remains at the top. And T' is a valid derivation because after removing E, literals in C are no longer present in the first place hence we can easily remove all the clauses of the type C' and C'_N.

■

3.2.2 Width and Size

In last section we saw width-size relation for tree-like resolution proofs. In this section we give a relation between $w(\vdash F)$ and $S(\vdash F)$ for general resolution proofs via Theorem 3.9. The Theorem was proved in [9]. Here we are presenting the proof from the survey, 'Proof Complexity' by Paul Beame [6].

Theorem 3.9. *[9] Let F be an unsatisfiable CNF formula over n variables. Then* $w(\vdash F) \leq w(F) + O(\sqrt{n \ln S(\vdash F)})$

Proof. If $S(\vdash F) = 1$. Then there is a resolution proof of size 1 with only the empty clause. Hence its width is $0 \leq w(F) + 0$ and we are done. Thus assume that $S(\vdash F) > 1$. Let π be a resolution proof of F with weakening rule of smallest size and define $d = \lceil \sqrt{2n \ln S(\vdash F)} \rceil$. If $d \geq 2n$ then we already have $w(\vdash F) \leq d < 2d + w(F) + 1$ (since any clause can have at most $2n$ distinct literals). Else $d < 2n$, and define $a = \left(1 - \frac{d}{2n}\right)^{-1}$. Let us call a clause in π large if its width is at least d and let π^* be the set of large clauses in π.

Consider the following statement:

$Stmt(m)$: Given any F' an unsatisfiable CNF formula over m variables and π' a resolution proof of F' with weakening rule. Let Q be the number of large clauses in π'. Then $\forall b$ if $Q < a^b$ then $w(\vdash F') \leq d + w(F') + b$.

We will prove $\forall m \leq n, Stmt(m)$ using induction on m. But before proving this let us first show how this proves Theorem 3.9: $Stmt(m)$ is true $\forall m \leq n$. In particular $Stmt(n)$ is true. Therefore for an unsatisfiable CNF formula F over n variables and for π a smallest resolution proof of F, fix $b = \lfloor \log_a |\pi^*| \rfloor + 1$. Clearly b satisfies $|\pi^*| < a^b$. Therefore from $Stmt(n)$ we have

$$\begin{aligned} w(\vdash F) &\leq d + w(F) + b \\ &= d + w(F) + \lfloor \log_a |\pi^*| \rfloor + 1 \\ &\leq d + w(F) + d + 1, \text{ by Claim 3.10 (see below)} \\ &= 2d + 1 + w(F) \\ &= O(\lceil \sqrt{2n \ln S(\vdash F)} \rceil) + w(F) \end{aligned}$$

Now we state and prove Claim 3.10 which completes the above arguments.

The way a and d are defined the following claim holds:

Claim 3.10. $\lfloor \log_a |\pi^*| \rfloor \leq \lceil \sqrt{2n \ln S(\vdash F)} \rceil = 1 \cdot d.$

Proof. We know that $\ln(1-x) \leq -x, \text{ for } x < 1$. Put $x = d/2n$. We have

$$\ln\left(1 - \frac{d}{2n}\right) \leq -\frac{d}{2n}$$
$$-\ln\left(1 - \frac{d}{2n}\right) \geq \frac{d}{2n}$$

Therefore we have,

$$\ln a \geq d/2n, \text{ since } a = \left(1 - \frac{d}{2n}\right)^{-1} \tag{3.2}$$

Now we are ready to prove the claim. We have

$$\begin{aligned}
\lfloor \log_a |\pi^*| \rfloor &\leq \log_a |\pi^*| \\
&= \frac{\ln |\pi^*|}{\ln a} \\
&\leq \frac{\ln |\pi|}{\ln a}, \text{ as } |\pi^*| \leq |\pi| \\
&\leq \frac{\ln |\pi|}{d/2n}, \text{ by Equation 3.2} \\
&\leq \frac{\lceil 2n \ln |\pi| \rceil}{d} \\
&= \frac{d^2}{d}, \text{ as } |\pi| = S(\vdash F) \\
&= d
\end{aligned}$$

■

Now we give a proof of $Stmt(m), \forall m \leq n$. The prove is by induction on m.

Base case When $m = 0, b \geq 0$, then $S(\vdash F') = 1$ and we are done as above.

Induction hypothesis $\forall G \in UNSAT$ with $m_1 < m \leq n$ variables. Let π'' be a resolution proof of G with weakening rule and Q' be the number of large clauses in π''. Then assume that $Stmt(m_1)$ holds. That is, $\forall b$ if $Q' < a^b$ then $w(\vdash G) \leq d + w(G) + b$.

Induction step We are given an unsatisfiable CNF formula F' with $m \leq n$ variables and a resolution proof π' with weakening rule. Let Q be the number of large clauses in π'. We need to show that $\forall b$ if $Q < a^b$ then $w(\vdash F') \leq d + w(F') + b$. Thus consider any b such that $Q < a^b$. And we will show that $w(\vdash F') \leq d + w(F') + b$ holds. When $n \geq m \geq 1$ and $b = 0$ then $Q < 1$ implies π' has no large clauses and hence $w(\pi') < d < 2d + 1 + w(F') = 2\lceil \sqrt{2n \ln S(\vdash F')} \rceil + 1 + w(F')$. Since $w(\vdash F')$

is minimum over all proofs, the claim holds. Thus assume that $b > 0$. Let T be the corresponding DAG for π'. See Figure 2(a).

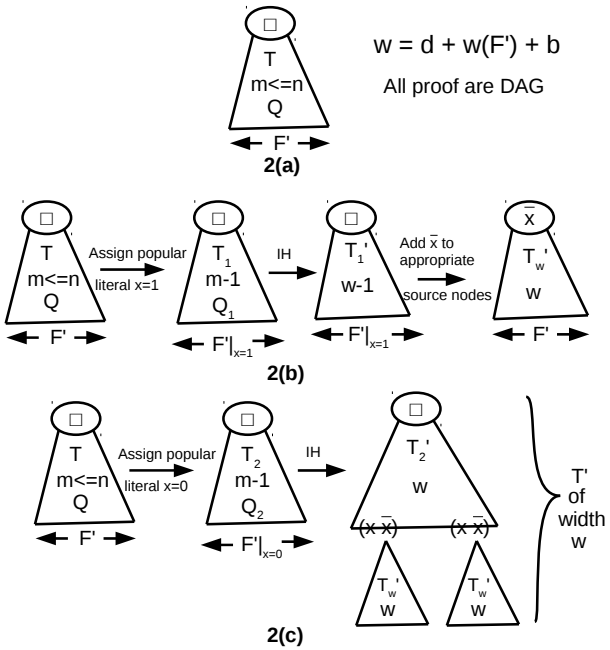

Figure 2: (a) General resolution proof T with weakening rule of F' over m <= n variables. T has Q large clauses. (b) Assign popular literal x=1 in T to get T_1. T_1 is a resolution proof with weakening rule of $F'|_{x=1}$ with Q_1 large clauses. Using T_1 and induction hypothesis construct a proof of F' or derivation of $\bar{x}$ with width at most w. (c) If (b) fails to construct a desired proof then assign popular literal x=0 in T to get T_2. Using T_2, induction hypothesis and T_w' construct a proof T' with weakening rule of F' with width at most w.

The idea behind this proof is to repeatedly find the popular literals appearing in large clauses of T. Resolving on these literals at the very beginning allows us to keep the width of the whole proof small.

Observe that F' contains at most $2m$ literals and at least d of them appears in any large clause, hence there exists a literal say x which occurs in at least $\frac{d}{2m}$ fractions of large clauses. (To verify the above statement, consider a matrix A, whose rows corresponds to large clauses and columns corresponds to literals. Thus A has Q rows and at most $2m$ columns. Define $A[i][j] = 1$ iff j^{th} literal belongs to i^{th} large clause. Clearly, each row of A has at least d $1's$ and total number of $1's$ by adding row-wise in A is at least $d \cdot Q$. Let each column of A has at most M $1's$. Then we have, $M \cdot 2m \geq d \cdot Q$, which implies that $M \geq \frac{d}{2m} Q$).

Choose the literal x that occurs most frequently in large clauses and set $x = 1$ in T to get T_1. That is, from T construct a DAG T_1 as follows: $\forall C \in T$, if $x \in C$ then $C \leftarrow 1$ and if $\neg x \in C$ then $C \leftarrow C \backslash \{\neg x\}$. After this restriction, remove all the 1 clauses from T along with their incidence edges to get T_1.

We claim that T_1 is a resolution proof with weakening rule for the formula $F'|_{x=1}$: all the source vertices of T_1 trivially belongs to $F'|_{x=1}$ and also the empty clause belongs to T_1 at the top. Thus it only remains to show that all internal vertices of

T_1 are derived either by resolution rule or by weakening rule. There are 4 cases (see Figure 3):

Case 1 In T, if there is a clause $E = (C \vee D)$ which is derived from the clauses $(C \vee \neg x)$ and $(D \vee x)$ by resolving x. Then in T_1 we have the corresponding clause $E_1 = (C \vee D)$ with only one child (C) which is a valid weakening rule. See Figure 3(a).

Case 2 In T if there is a clause $E = (\neg x \vee C \vee D)$ derived from the clauses $(\neg x \vee C \vee y)$ and $(D \vee \neg y)$ by resolving y. Then in T_1 we have $E_1 = (C \vee D)$ with children $(C \vee y)$ and $(D \vee \neg y)$. A valid resolution step. See Figure 3(b).

Case 3 In T if there is a clause $E = (x \vee C \vee D)$ derived from the clauses $(x \vee C \vee y)$ and $(D \vee \neg y)$ by resolving on y. Then in T_1 only the clause $(D \vee \neg y)$ remains. Hence nothing to prove. See Figure 3(c).

Case 4 In T if there is a clause $E = (C \vee D \vee \neg x)$ derived from the clause $(C \vee \neg x)$ by the weakening rule. Then in T_1 we have left with $E_1 = (C \vee D)$ with only child (C), a valid weakening rule. See Figure 3(d). Similarly if we have x in place of $\neg x$ then in T_1 we have removed both the clauses hence nothing to prove.

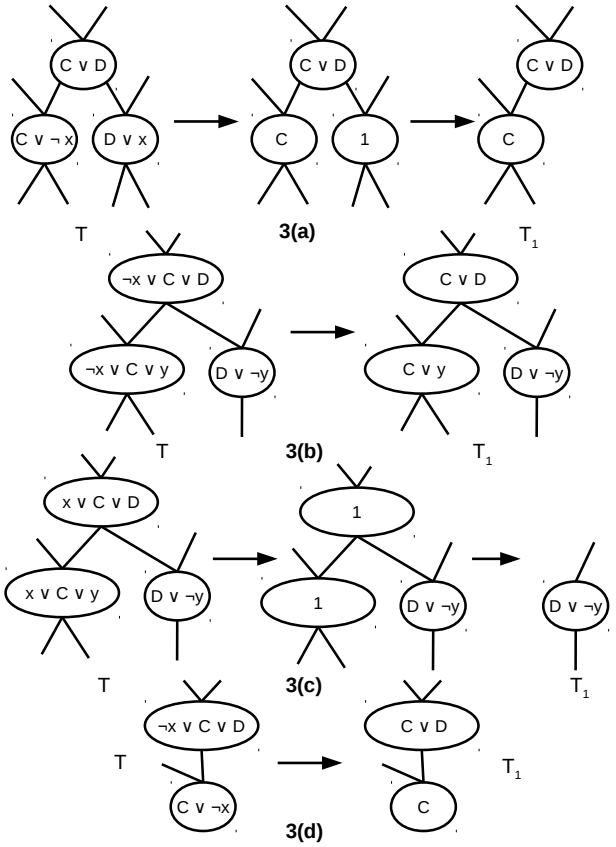

Figure 3: Showing different cases that arises in proving that T_1 is a valid resolution proof of $F'|_{x=1}$ with weakening rule.

Observe that as x is a popular literal it satisfies $\frac{d}{2m}$ fractions of large clauses of T. Hence in T_1 the number of large clauses left is

$$\begin{aligned} Q_1 &\leq Q\Big(1 - \frac{d}{2m}\Big) \\ &\leq Q\Big(1 - \frac{d}{2n}\Big), \text{ as } m \leq n \\ &= Q/a \\ &< a^{b-1}, \text{ as } Q < a^b \end{aligned}$$

From induction hypothesis $Stmt(m-1)$ is true. Thus for an unsatisfiable CNF formula $F'|_{x=1}$ with $m-1$ variables and T_1 a resolution proof of $F'|_{x=1}$ with weakening rule with Q_1 number of large clauses, we know that $\forall b$ if $Q_1 < a^b$ then $w(\vdash F'|_{x=1}) \leq d + w(F'|_{x=1}) + b$. In particular we know that for $b-1$, $Q_1 < a^{b-1}$ is true. Therefore there must be a resolution proof T_1' of $F'|_{x=1}$ with weakening rule having width at most $d + w(F'|_{x=1}) + b - 1 \leq d + w(F') + b - 1$.

Let $F'_{\neg x} \subseteq F'$ be the subset of all clauses containg $\neg x$ and let $F''_{\neg x} \subseteq F'_{\neg x}$ be the subset of clauses which are used as initial clauses in T_1' with $\neg x$ being removed. From the proof T_1' construct a derivation T'_w by adding back $\neg x$ in all the initial clauses $F''_{\neg x}$ and propogate them up by exactly the steps taken in T_1'. Since there is no resolution step which resolves on x, T'_w is a valid derivation. Moreover the width of T'_w increases by 1. There are two possibilities: either the empty clause or $\neg x$ is on the top of T'_w. In the first case T'_w is a resolution proof of F' with weakening rule having width $w(T'_w) \leq d + w(F') + b$ which implies $w(\vdash F') \leq d + w(F') + b$ and we are done. Otherwise we have a derivation T'_w of $\neg x$ with weakening rule from F' having width $\leq d + w(F') + b$. See Figure 2(b).

Repeat the procedure by setting $x = 0$ on T to get T_2. That is construct a DAG T_2 from T as follows: $\forall C \in T$, if $\neg x \in C$ then $C \leftarrow 1$ and if $x \in C$ then $C \leftarrow C \backslash \{x\}$. After the restriction remove all 1 clauses along with their incidence edges to get T_2. By the similar argument T_2 is a resolution proof of $F'|_{x=0}$ with weakening rule. The number of large clauses left in T_2 is:

$$\begin{aligned} Q_2 &\leq Q \\ &< a^b, \text{ as } Q < a^b \end{aligned}$$

Again by induction hypothesis $Stmt(m-1)$ is true. Therefore for the unsatisfiable CNF formula $F'|_{x=0}$ with $m-1$ variables and T_2 a resolution proof of $F'|_{x=0}$ with weakening rule with Q_2 number of large clause, we have for b, $Q_2 < a^b$ is true, therefore there must be a resolution proof T_2' of $F'|_{x=0}$ with weakening rule having width $d + w(F'|_{x=0}) + b \leq d + w(F') + b$.

Now we use the same trick as in Theorem 3.6. Let $F'_x \subseteq F'$ be the subset of clauses of F' containing x and let $F''_x \subseteq F'_x$ be the subset of initial clauses used in T_2' with x being removed. Using T_2' and T'_w construct a resolution proof T' of F' with weakening rule such that $w(T') \leq d + w(F') + b$ as follows: add x in all the clauses F''_x of T_2' and resolve them before the leaf level of T_2' by using the copy of T'_w which is a derivation of $\neg x$. See Figure 2(c). After this we again get back F''_x which can be used by T_2' to derive an empty clause. Since $w(T'_w), w(T_2') \leq d + w(F') + b$ it

implies that $w(T') \leq d + w(F') + b$. Since $w(\vdash F')$ is minimum over all proof, the claim follows.

■

Remark 3.11. *T' is a resolution proof of F' with weakening rule and width at most $d+w(F')+b$. From T' we may obtain a resolution proof T'' of F' without weakening rule and without increasing its width. For this, first convert T' into a tree-like resolution proof T_3 of F' with weakening rule, just by rederiving all the clauses when needed. This makes sure that every clause is used at most once in any inference rule. Observe that this step does not increase the width of the proof. Now apply Lemma 2.3 on T_3 to obtain a tree-like resolution proof T'' of F' without weakening rule. Observe that the width of T'' has not been increased as while eliminating a weakening rule Lemma 2.3 does not increase the width of the given proof.*

Corollary 3.12. *For an unsatisfiable CNF formula F over n variables, we have $S(\vdash F) = e^{\Omega\left(\frac{(w(\vdash F)-w(F))^2}{n}\right)}$.*

Proof. From theorem 3.9, we have,

$$\begin{aligned} w(\vdash F) &\leq w(F) + O(\sqrt{2n \ln S(\vdash F)}) \\ w(\vdash F) - w(F) &\leq 2\sqrt{2n \ln S(\vdash F)} \\ \frac{(w(\vdash F) - w(F))^2}{4n} &\leq \ln S(\vdash F) \\ \Omega\Big(\frac{(w(\vdash F) - w(F))^2}{n}\Big) &= \ln S(\vdash F) \\ e^{\Omega\left(\frac{(w(\vdash F)-w(F))^2}{n}\right)} &= S(\vdash F) \end{aligned}$$

■

3.3 Tree-like vs General Resolution Proofs

The next question is, can the width-size relationship, proved in Corollary 3.12 be improved? M. L. Bonet and N. Galesi in their paper [11] came up with a negative answer to this problem by proving the following theorem. Theorem 3.13 is also used to show that there are formulas with exponential gap in the size of tree-like and general refutation. In this section we present the proof of Theorem 3.13.

Theorem 3.13. *[11] There is a family of formulas $\{F_n\}$ in n variables with constant clause width that have polynomial-size resolution proofs but require resolution proof width $\Omega(\sqrt{n})$*

Before proving Theorem 3.13 let us see how it show that Corollary 3.12 is tight.

From Corollary 3.12 we know that for all unsatisfiable CNF formula F over n variables we have

$$S(\vdash F) = e^{\Omega\left(\frac{(w(\vdash F)-w(F))^2}{n}\right)}$$

And from Theorem 3.13 we know that we have an unsatisfiable CNF formula F' over n variables such that $S(\vdash F') = n^{O(1)}$, $w(\vdash F') = \Omega(\sqrt{n})$ and $w(F') = c$, where c is some constant (we will see that actually $c = 3$).

For contradiction let us suppose that there is some improvement on Corollary 3.12. That is for all unsatisfiable CNF formula F over n variables we have

$$S(\vdash F) = e^{\Omega\left(\frac{(w(\vdash F)-w(F))^2}{n^{1-\epsilon}}\right)}, \text{ for some } \epsilon > 0 \tag{3.3}$$

In particular this must hold for F' as well. Thus from Theorem 3.13 and Equation 3.3 we have

$$\begin{aligned}
n^{O(1)} &= e^{\Omega\left(\frac{(\sqrt{n}-3)^2}{n\cdot n^{-\epsilon}}\right)} \\
e^{O(1)\ln n} &= e^{\Omega\left(\frac{n+9-6\sqrt{n}}{n\cdot n^{-\epsilon}}\right)} \\
e^{O(1)\ln n} &= e^{\Omega\left(n^{\epsilon}(1+\frac{9}{n}-\frac{6}{\sqrt{n}})\right)} \\
O(1)\ln n &\geq n^{\epsilon}\left(1+\frac{9}{n}-\frac{6}{\sqrt{n}}\right) \\
c'\cdot\frac{\ln n}{n^{\epsilon}} &\geq \left(1+\frac{9}{n}-\frac{6}{\sqrt{n}}\right), \text{ where } c' \text{ is some constant}
\end{aligned}$$

Since $\epsilon > 0$, as $n \to \infty$, $c' \cdot \frac{\ln n}{n^{\epsilon}} \to 0$. However, as $n \to \infty$, $\left(1 + \frac{9}{n} - \frac{6}{\sqrt{n}}\right) \to c_1$, where c_1 is some constant > 0. Contradiction.

Now we start presenting the proof of Theorem 3.13. To prove Theorem 3.13, Bonet and Galesi [11] considered the CNF formula DGT_n expressing the negation of the following graph property:

Graph Property (GP) In any simple directed graph with n vertices, closed under transitivity, and with no cycles of size two (i.e, no 2-cycles), there is a source vertex.

They showed that a modification of DGT_n formula verifies the desired properties. First we will show that (GP) holds.

Graph theoretic proof of (GP) Let $G = (V, E)$, with $|V| = n$ be a simple directed graph, closed under trasitivity and with no 2-cycles.We will prove by induction on n that G has a source vertex.

Base case When $(n = 1)$, we have $|E| = 0$ and the unique vertex is a source vertex.

Induction hypothesis Suppose every simple directed graph with less than $n > 1$ vertices, which preserves transitivity and with no 2-cycles, has a source vertex.

Induction step Now we have $G = (V, E)$, with $|V| = n$. For contradiction, suppose G has no source vertex. Consider a graph G' obtained from G by deleting a vertex v along with its incident edges. Clearly G' preserves transitivity and has no 2-cycles, hence by induction hypothesis G' has a source vertex. Let $s_1, \ldots, s_j$ be

the source vertices of G'. Now consider the graph $G = G' \cup v$. If outdegree of v is zero then clearly source vertices of G' remains the source vertices of G as well. Hence we have $\{(v, s_i) \in E | 1 \leq i \leq j\}$. As v is not a source vertex, $\exists v_1$, such that, $\{(v_1, v) \in E\}$, and so on. See Figure 4.

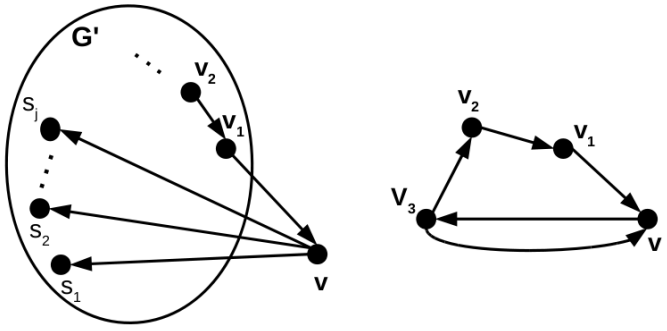

Figure 4: (a) Induction step of the proof: $s_1,\ldots,s_j$ are the source vertices of G'. As there are no source vertex in G, consider the reverse path from v. (b) Shows a case where the reverse path ends in v. As there is a path from v_3 to v, due to transitivity there is an edge between v_3 to v which completes a 2-cycle.

Now consider the reverse path $v, v_1, v_2, \ldots$. As G is a finite graph, this reverse path has to end at some vertex. The end vertex can be v or a source vertex of G' or some v_i. In all the cases we have a cycle and due to transitivity we get a 2-cycle, which is a contradiction. This proves (GP).

Encoding $\neg GP$ as an unsatisfiable CNF formula We can easily encode $\neg GP$ as an unsatisfiable CNF formula DGT_n over variables $x_{i,j}, i, j \in [n], i \neq j$, which is supposed to be assigned "True" iff there is an edge from vertex i to vertex j. DGT_n contains the following clauses:

(1) **Transitivity.** $(\neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k}), \quad i, j, k \in [n], i \neq j \neq k$. This clause encodes that if there is an edge from vertex i to vertex j, and vertex j to vertex k, then there must be an edge from vertex i to vertex k.

(2) **No 2-cycles.** $(\neg x_{i,j} \vee \neg x_{j,i}), \quad i, j \in [n], i \neq j$. This clause encodes that there are no 2-cycles.

(3) **No source vertices.** $\bigvee_{k=1,k\neq j}^{n} x_{k,j} \quad j \in [n]$. This clause encodes that there are no source vertices.

As (GP) is true, there are no assignments which can simultaneously make all the above clauses true. To be precise, every assignment to the variable of DGT_n corresponds to a simple directed graph on n vertices. And every satisfying assignment of DGT_n corresponds to a simple directed graph which preserves transitivity, has no 2-cycles, and has no source vertices. We know because of (GP) that there are no such simple directed graphs. Thus DGT_n is unsatisfiable.

There are $\binom{n}{2}$ boolean variables and $\binom{n}{3}3!, \binom{n}{2}, n$ clauses of type $(1), (2)$, and (3) respectively. The width of DGT_n is $n-1$ (i.e, $w(DGT_n) = n-1$), due to

the clauses of type (3). However, in order to prove Theorem 3.13 we need a formula with constant initial width. To achieve this we introduce n new variables $y_{0,j}, \ldots, y_{j-1,j}, y_{j+1,j}, \ldots, y_{n,j}$ for each $j \in [n]$ and replace the clauses in (3) by the following clauses:

$(3')$ $\quad \neg y_{0,j} \wedge \bigwedge_{i=1,i\neq j}^{n} (y_{i-1,j} \vee x_{i,j} \vee \neg y_{i,j}) \wedge y_{n,j}$

We call the modified formula as $MDGT_n$. Clearly $(3')$ has $n(n+1)$ clauses. Thus there are total $\left(\binom{n}{3}3! + \binom{n}{2} + n(n+1)\right)$ number of clauses in $MDGT_n$ and $MDGT_n$ has $\binom{n}{2}$ x-variables and n^2 y-variables. Thus in total $MDGT_n$ has $\Theta(n^2)$ variables. $MDGT_n$ remains unsatisfiable CNF formula. This is because for each $j \in [n]$, if a clause in (3) evaluates to 0 by an assignment α_x of x variables, then for all possible assignments α_y of y-variables the clause in $(3')$ for the corresponding j, when evaluated in $\alpha_x + \alpha_y$ gives a 0. Moreover, it can be seen that for each $j \in [n]$, if a clause in (3) evaluates to 1 by any assignment α_x of x-variable then there exists assignments α_y of y-variables such that the corresponding $(3')$ clause when evaluated on $\alpha_x + \alpha_y$ also gives a 1.

Thus we have an unsatisfiable CNF formula $MDGT_n$ over $\Theta(n^2)$ variables and with $w(MDGT_n) = 3$. Now we will show that $MDGT_n$ has a polynomial size general resolution refutation.

Theorem 3.14. *[11] There exists a general resolution proof π of $MDGT_n$ such that $|\pi| = n^{O(1)}$. That is, $S(\vdash MDGT_n) \in n^{O(1)}$.*

Proof. For each vertex $j \in [n]$ we have a clause in (3), that is, $\bigvee_{k=1,k\neq j}^{n} x_{k,j}$. Let us call this clause as $3j$. Simlilarly we have CNF $3'j$. First part of the proof is to obtain $3j$ from $3'j$ for each $j \in [n]$ by eliminating y-variables using resolution rule. This can be achieved simply as follows: For each $j \in [n]$, from $3'j$, go on resolving variables in this order: $y_{0,j}, y_{1,j}, \ldots, y_{j-1,j}, y_{j+1,j}, \ldots, y_{n,j}$ to get $3j$. More formally we have,

$$\dfrac{\dfrac{\dfrac{(\neg y_{0,j}) \quad (y_{0,j} \vee x_{1,j} \vee \neg y_{1,j})}{(x_{1,j} \vee \neg y_{1,j})}\, y_{0,j} \quad (y_{1,j} \vee x_{2,j} \vee \neg y_{2,j})}{(x_{1,j} \vee x_{2,j} \vee \neg y_{2,j})}\, y_{1,j} \quad (y_{2,j} \vee x_{3,j} \vee \neg y_{3,j})}{\vdots}\, y_{2,j}$$
$$\dfrac{\vdots}{(x_{1,j} \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee \ x_{n,j})}\, y_{n,j}$$

Observe that this part of the proof is actually a tree-like proof of size $2n$ for each $j \in [n]$. Thus overall the first part has size quadratic in n and derives DGT_n from $MDGT_n$.

For the second part of the proof we start with the unsatisfiable CNF formula DGT_n. We will now give a general resolution refutation of DGT_n. Let us define,

$$C_m(j) = \bigvee_{i=1,i\neq j}^{m} x_{i,j} \qquad j, m \in [n].$$

and

$$\mathcal{C}_m = \bigwedge_{j=1}^{n} C_m(j) \qquad m \in [n].$$

The interpretation of $C_m(j)$ for some $j, m \in [n]$ is $-$ "vertex j has an incoming edge from at least one vertex in $[m]$. Similarly the interpretation of $\mathcal{C}_m$ for some $m \in [n]$ is $-$ "each vertex j has an incoming edge from at least one vertex in $[m]$.

Proof idea We will show by downward induction from n to 2 that $\mathcal{C}_k, 2 \leq k \leq n$ is true. At the k^{th} step we will derive $C_{k-1}(j)$ for each $j = 1, \ldots, n$ using the initial clauses and the clauses $C_k(j)$ and $C_k(k)$ obtained at the previous step. This will eventually derive $\mathcal{C}_{k-1}$. At the end we have,

$$\mathcal{C}_2 = \bigwedge_{j=1}^{n} C_2(j) = (x_{2,1}) \wedge (x_{1,2}) \wedge (x_{1,3} \vee x_{2,3}) \wedge \cdots \wedge (x_{1,n} \vee x_{2,n})$$

Now we can easily derive an empty clause from $\mathcal{C}_2$ by using an initial clause of type (2) with $i = 1$ and $j = 2$. That is,

$$\dfrac{\dfrac{(\neg x_{1,2} \vee \neg x_{2,1}) \quad (x_{2,1})}{(\neg x_{1,2}) \quad (x_{1,2})}\, x_{2,1}}{\square}\, x_{1,2}$$

Base case: $\mathcal{C}_n = \bigwedge_{j=1}^{n} C_n(j) = \bigwedge_{j=1}^{n} \vee_{i=1, i\neq j}^{n} x_{i,j}$, is indeed (3) which we have already derived in first part from (3′). Done.

Induction hypothesis: Suppose we have derived $C_k(j) = \bigvee_{i=1, i\neq j}^{k} x_{i,j} = (x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j})$ for $j = 1, \ldots, n$ and hence $\mathcal{C}_k$.

Induction step: Now we have $k \in [n] \backslash \{1, 2\}$. We will show how to derive in two steps $C_{k-1}(j)$ for a value $j \in [n]$ using initial clauses of type (1) (i.e, transitivity), type (2) (i.e, no 2-cycles) and the clauses $C_k(j), C_k(k)$.

First step (a)**:** Perform in parallel the following resolution steps, each one resolving the variable $x_{k,j}$:

$$(1)a \quad \frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j}) \quad (\neg x_{1,k} \vee \neg x_{k,j} \vee x_{1,j})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{1,k})}$$

$$(2)a \quad \frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j}) \quad (\neg x_{2,k} \vee \neg x_{k,j} \vee x_{2,j})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{2,k})}$$

$\cdots$

$$(j-1)a \quad \frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j}) \quad (\neg x_{j-1,k} \vee \neg x_{k,j} \vee x_{j-1,j})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{j-1,k})}$$

$$(j)a \quad \frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j}) \quad (\neg x_{j,k} \vee \neg x_{k,j})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{j,k})}$$

$$(j+1)a \quad \frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j}) \quad (\neg x_{j+1,k} \vee \neg x_{k,j} \vee x_{j+1,j})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{j+1,k})}$$

...

$$(n)a \quad \frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k,j}) \quad (\neg x_{n,k} \vee \neg x_{k,j} \vee x_{n,j})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{n,k} \vee x_{n,j})}$$

Second step (b)**:** $C_{k-1}(j)$ is obtained by the following refutation in which we are resolving along the variables $x_{1,k}, x_{2,k}, \ldots, x_{k-1,k}$:

$(1)b$ Resolve $x_{1,k}$ from the resulting clause of $(1)a$ and $C_k(k)$. That is,

$$\frac{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{1,k}) \quad (x_{1,k} \vee \ldots, \vee x_{k-1,k})}{(x_{1,j} \vee \cdots \vee x_{j-1,j} \vee x_{j+1,j} \vee \cdots \vee x_{k-1,j} \vee x_{2,k} \vee \cdots \vee x_{k-1,k})}$$

$(2)b$ Resolve $x_{2,k}$ from the resulting clause of $(2)a$ and $(1)b$. That is,

$$\frac{(x_{1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{2,k}) \quad (x_{1,j} \vee \cdots \vee x_{k-1,j} \vee x_{2,k} \vee \cdots \vee x_{k-1,k})}{(x_{1,j} \vee \cdots \vee x_{k-1,j} \vee x_{3,k} \vee \cdots \vee x_{k-1,k})}$$

...

$(k-1)b$ Resolve $x_{k-1,k}$ from the resulting clause of $(k-1)a$ and $(k-2)b$. That is,

$$\frac{(x_{1,j} \vee \cdots \vee x_{k-1,j} \vee \neg x_{k-1,k}) \quad (x_{1,j} \vee \cdots \vee x_{k-1,j} \vee x_{k-1,k})}{(x_{1,j} \vee \cdots \vee x_{k-1,j})}$$

The resulting clause of $(k-1)b$ is exactly $C_{k-1}(j)$.

Size of the refutation At the k^{th} induction step for deriving $C_{k-1}(j)$ for some $j \in [n]$, in the first and the second step we are using at most n resolution rules. Thus total of at most $2n$ resolution rules. As to derive $\mathcal{C}_{k-1}$ we need to derive $C_{k-1}(j)$ for each $j \in [n]$ hence at the k^{th} induction step we are using at most $\Theta(n^2)$ resolution rules. There are $n-2$ induction steps and thus the total resolution rules used by the second part of the proof is $\Theta(n^3)$.

Thus in first part we have used at most n^2 resolution rules and in the second part $\Theta(n^3)$ resolution rules. Hence the size of the refutation of $MDGT_n$ is bounded by $\Theta(n^3)$. We have bounded the size of the refutation in terms of the number of vertices n in the given directed graph. We will now bound the size in terms of the number of variable in $MDGT_n$. We know that $MDGT_n$ has $\Theta(n^2)$ variables. Let $N = cn^2$ be the number of variables in $MDGT_n$. Thus size of the refutation is bounded by $\Theta(n^3) = \Theta(N^{3/2})$. ■

Till now we have shown that $MDGT_n$ which has $\Theta(n^2)$ variables has constant clause width and has polynomial size general resolution refutation. Now in the next theorem we will prove that any resolution refutation of $MDGT_n$ contains a clause having at least $\Omega(n)$ variables.

Theorem 3.15. *[11] Any resolution proof of $MDGT_n$ must have a clause of width $\Omega(n)$. That is, $w(\vdash MDGT_n) \in \Omega(n)$.*

Recall as discussed in the proof of Theorem 3.1 (lower bound for PHP^n_{n-1}), given any resolution refutation one can test its correctness by assigning values to the variables. A correct resolution proof shows that no assignments can satisfy all the initial set of clauses. Consider refutations that only show that a certain subset of assignments (i.e, critical assignments, yet to be defined) can not satisfy all the

initial clauses simultaneously. In other words, if we substitute any assignments from this subset, the refutation correctly produces a 0-path. For assignments outside this subset it may or may not produces a 0-path. Thus this has relaxed the notion of resolution refutation. However proving width lower bound for this relaxed refutations also apply to the general refutations. That is, proving that any relaxed refutation contains a clause with width $\Omega(n)$ implies that general refutation must also contains a clause with width $\Omega(n)$.

Before proving width lower bound for relaxed refutation we need to define critical assignment along with some other notations.

Critical assignment for $MDGT_n$ First we define critical assignment for the formula DGT_n. We call an assignment of DGT_n a critical assignment if it satisfies all the clauses of DGT_n except one clause of type (3). If the only falsified clause is $3j$ for some $j \in [n]$ then we call such a critical assignment a j-critical assignment. As already pointed out, any assignment for the variables of DGT_n corresponds to a directed graph. For a j-critical assignment the corresponding graph is a linear directed acyclic graph on n vertices, closed under transitivity and with vertex j as the only source vertex. This is because $x_{i,j} = 1$ iff there is a directed edge (i, j) in the graph and such a linear directed graph is closed under transitivity, has no 2-cycles, and every vertex except the vertex j has a predecessor.

A j-critical assignment for $MDGT_n$ is defined the following way: first give a j-critical assignment α_j for DGT_n (i.e, only for x's variables). Clearly α_j satisfies all the initial clauses of DGT_n except $3j$. Extend α_j by assigning values to y-variables in such a way as to make false only the formula $3'j$ and true all the other $3'k, k \neq j$ As we have pointed out earlier (see before Theorem 3.14) this is always possible. A resulting assignment β_j is a j-critical assignment for $MDGT_n$.

Some more notations For $j \in [n]$, let $N_j = \bigwedge_{i=1, i\neq j}^{n} (\neg x_{i,j} \vee \neg x_{j,i})$. N_j is satisfied by an assignment iff vertex j does not belong to any 2-cycles in the corresponding directed graph. Thus CNF formula N_j is satisfied by all critical assignments. Consider the formula $\mathcal{C}_j = N_j \wedge 3'j$. Thus $\mathcal{C}_j$ is satisfied by an assignment iff vertex j does not belong to any 2-cycles and is not a source vertex in the corresponding directed graph. Let $Vars(j)$ be the set of variables of $\mathcal{C}_j$. Observe that $Vars(j)$ contains all the variables of $MDGT_n$ that refer to the vertex j. Also observe that the formula $\mathcal{C}_j$ is not satisfied by any j-critical assignments, but is satisfied by any i-critical assignments with $i \neq j$.

Now we are ready to prove Theorem 3.15 by proving $\Omega(n)$ width lower bound for any relaxed refutation of $MDGT_n$.

Proof of Theorem 3.15 .
The idea behind this proof is as follows: given any relaxed refutation π of $MDGT_n$, we will define a complexity measure μ for each clause $C \in \pi$. We will show that $\exists C \in \pi$ such that $n/3 \leq \mu(C) \leq 2n/3$ and using this fact we will show that for this C we have, $width(C) \geq n/6$. This will show that $w(\pi) = \max_{C\in\pi}\{width(C)\} \geq n/6$. And since π can be any relaxed refutation, this will imply an $\Omega(n)$ lower bound on any relaxed refutation of $MDGT_n$.

Complexity measure μ For each $I \subseteq [n]$, let $\mathcal{C}_I = \bigwedge_{i \in I} \mathcal{C}_i$. For any $C \in \pi$, define $\mu(C)$ as the size of the minimal $I \subseteq [n]$ such that all critical assignments satisfying $\mathcal{C}_I$ also satisfy C. In other words, let P_1 be the following property:

$$P_1 \equiv \forall \text{critical assignments } \beta, \beta(\mathcal{C}_I) \implies \beta(C)$$

Then, for $C \in \pi$,

$$\mu(C) = \min_{I \subseteq [n]} \{|I| : I \text{ satisfies } P_1\}$$

As $\mathcal{C}_{[n]}$ is unsatisfiable $\forall C \in \pi, \mu(C)$ is defined. Call the minimal $I \subseteq [n]$ such that $\mu(C) = |I|$ as a certificate for $\mu(C)$.

Properties of μ Let π be any relaxed refutation of $MDGT_n$. For any clause $C \in \pi$ we have,

1. $\mu(C) = 0$, if C is any initial clause of type (1) or (2). This is because any critical assignment satisfies initial clauses of type (1) and (2).

2. $\mu(C) = 1$, if C is some initial clause of type $3'i$. This is because a minimal $I \subseteq [n]$ for which $\mathcal{C}_I \implies C$ holds is when $I = \{j\}, j \neq i$. Since any j-critical assignment with $j \neq i$ satisfies the clause $3'i$. Thereby we have $\mu(C_i) = 1, i \in [n]$.

3. $\mu(\square) = n$. This is because $C_I \implies \square$ is true only for $I = [n]$ as only $C_{[n]}$ is unsatisfiable. For $I \neq [n]$, let $j \notin I$ then a j-critical assignment satisfied $\mathcal{C}_I$ but not $\square$.

4. μ is subadditive with respect to resolution rule. That is, for a resolution rule $\frac{A \quad B}{C}$, we have $\mu(C) \leq \mu(A) + \mu(B)$: let $J, K \subseteq [n]$ be the certificate for $\mu(A), \mu(B)$, respectively. Since we have $A \wedge B \implies C$ and for any critical assignment β we have $\beta(\mathcal{C}_J) \implies \beta(A), \beta(\mathcal{C}_K) \implies \beta(B)$ thereby, we also have $\beta(\mathcal{C}_{J \cup K}) \implies \beta(C)$. Hence $\mu(C) \leq |J \cup K| \leq |J| + |K| = \mu(A) + \mu(B)$.

Claim 3.16. *For any relaxed refutation π of $MDGT_n$. There exists a clause $C' \in \pi$ such that $n/3 \leq \mu(C') \leq 2n/3$.*

Proof. The proof is similar to that of the proof of Claim 3.4. However for completeness we present here once again. Consider the proof graph G_π of π (edges are from the conclusion to the two hypothesis). The root of G_π corresponds to the empty clause and leaves corresponds to the source clauses of $MDGT_n$. One can find C' with the following simple search algorithm in G_π: starting with the root vertex of G_π, go on descending towards source nodes by following the children with larger μ values, until we encouter for the first time a clause A with $\mu(A) < n/3$. The parent of A is C': as A is the first clause with smaller μ values, we have $\mu(C') \geq n/3$. Also among the two children say A and B we have, $\mu(A) \geq \mu(B)$. Thus $\mu(B) < n/3$, and due to property 4, $\mu(C') \leq n/3 + n/3 = 2n/3$. ■

Now we will prove by contradiction that $width(C') \geq n/6$. Suppose not. Then $width(C') < n/6$. Let $I \subseteq [n]$ be a certificate for $\mu(C')$. Thus we have, $|I| \in [n/3, 2n/3]$. Since $|I| \geq n/3$ and $width(C') < n/6$ the following claim holds:

Claim 3.17. *There exists at least an $l \in I$ such that no variables from $Vars(l)$ belongs to C'.*

Proof. Each variable $x_{i,j}$ belongs to two different sets $Vars(i)$ and $Vars(j)$. In the worst case all the variables in C' mention different vertices. By assumption we have $width(C') < n/6$, hence they collectively captures $< 2n/6$ different sets $Vars(\cdot)$. Since $|I| \geq n/3$, there is at least one index in I verifying the claim. ■

Thus we have a clause $C' \in \pi$ such that $\mu(C') \in [n/3, 2n/3]$, I a certificate for $\mu(C')$ and $l \in I$ such that no variables from $Vars(l)$ belongs to C'. Given such an l, I and C' we have the following Claim:

Claim 3.18. *There exists an l-critical assignment α such that,*

1. $\forall i \in I \setminus \{l\}, \alpha(\mathcal{C}_i) = 1$,
2. $\alpha(\mathcal{C}_l) = 0$, *and*
3. $\alpha(C') = 0$.

Since I is a certificate of C' we have

$$\forall \text{ critical assignment } \beta, \beta(\mathcal{C}_I) \implies \beta(C')$$

Also let $K = I \setminus \{l\}$. Then proving the following statement:

$$\forall \text{ critical assignment } \beta, \beta(\mathcal{C}_K) \implies \beta(C')$$

will give us a contradiction, since the above statement implies that $\mu(C') \leq |K| < |I|$. Contradiction. With this information we give a proof of Claim 3.18.

Proof of Claim 3.18 .
We will prove this by contradiction. Suppose such a critical assignment α does not exists. Then we have three cases:

First case $\forall$ critical assignment β we have $\beta(\mathcal{C}_K) = 0$. In this case

$$\forall \text{ critical assignment } \beta, \beta(\mathcal{C}_K) \implies \beta(C')$$

trivially holds. This implies that $\mu(C') \leq |K| < |I|$. Contradiction by minimality of I. Therefore there exists at least one critical assignment β such that $\beta(\mathcal{C}_K) = 1$.

Second case $\forall$ critical assignment β with $\beta(\mathcal{C}_K) = 1$ we have $\beta(\mathcal{C}_l) = 1$. This implies that

$\forall$ critical assignment β with $\beta(\mathcal{C}_K) = 0$ we have
$\beta(\mathcal{C}_K) \implies \beta(C')$ holds, and
$\forall$ critical assignment β with $\beta(\mathcal{C}_K) = 1$ we have $\beta(\mathcal{C}_l) = 1$
$\therefore \beta(\mathcal{C}_K \wedge \mathcal{C}_l) = 1$

$\therefore \beta(\mathcal{C}_I) = 1 \implies \beta(\mathcal{C}')$.
thus in this case we have
$\beta(\mathcal{C}_K) \implies \beta(\mathcal{C}_l) \implies \beta(\mathcal{C}_I) \implies \beta(\mathcal{C}')$ holds.

Thus this implies that $\mu(C') \leq |K| < |I|$. Contradiction by minimality of I. Therefore there exists at least one critical assignment β such that $\beta(\mathcal{C}_K) = 1$ and $\beta(\mathcal{C}_l) = 0$.

Third case $\forall$ critical assignment β we have

$$[\beta(\mathcal{C}_K) = 1 \wedge \beta(\mathcal{C}_l) = 0] \implies \beta(\mathcal{C}') = 1$$

Equivalently,

$$\beta(\mathcal{C}_K) = 0 \vee \beta(\mathcal{C}_l) = 1 \vee \beta(\mathcal{C}') = 1$$

Pick any critical assignment β.

(a) If $\beta(\mathcal{C}_K) = 0$ then $\beta(\mathcal{C}_K) \implies \beta(C')$ holds (as in the first case).

(b) If $\beta(\mathcal{C}_K) = 1 \wedge \beta(\mathcal{C}_l) = 1$ then $\beta(\mathcal{C}_K) \implies \beta(C')$ holds (as in the second case).

(c) If $\beta(\mathcal{C}_K) = 1 \wedge \beta(\mathcal{C}_l) = 0$ then $\beta(\mathcal{C}') = 1$ and hence $\beta(\mathcal{C}_K) \implies \beta(C')$ holds trivially.

Thus we have,

$$\forall \text{ critical assignment } \beta, \beta(\mathcal{C}_K) \implies \beta(C') \text{ holds}$$

This implies that $\mu(C') \leq |K| < |I|$. Contradiction.

Therefore there must be a critical assignment α which satisfies all the three conditions of the Claim. Since α satisfies condition (1) and (2) of the Claim, it must be an l-critical assignment. ■

Fix such an l-critical assignment α. Thus we have $\forall i \in I \setminus \{l\}, \alpha(\mathcal{C}_i) = 1, \alpha(\mathcal{C}_l) = 0$, and $\alpha(\mathcal{C}') = 0$.

Define $J = [n] \setminus I$. As $|I| \leq 2n/3$, we have $|J| \geq n - 2n/3 = n/3$. Therefore by replacing I by J in claim 3.17 we deduce that there is at least one index $j \in J$ such that no variable from $Vars(j)$ appears in C'. By construction $I \cap J = \phi$ therefore $j \neq l$.

Now using the fact that variables from $Vars(l), Vars(j)$ do not belongs to C' we construct a j-critical assignment β from the l-critical assignment α such that $\forall i \in I, \beta(\mathcal{C}_i) = 1$ and $\beta(C') = 0$ which contradicts the assumption that I is a certificate for $\mu(C')$.

Construction of j-critical assignment β from l-critical assignment α
We know that α is a linear directed graph which preserves transitivity, has no cycles and with a single source vertex l. The idea is to get another linear directed graph from α just by placing the vertex j in front of the vertex l without distroying any property. we call it the j-critical assignment β. See Figure 5.

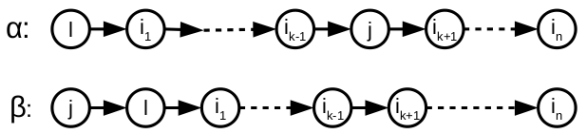

Figure 5: α is a linear directed graph which preserves transitivity, has no cycles and with a single source vertex l. For simplicity transitive edges are omitted.
β is a linear directed graph obtained from α just by placing j infront of l. β receives all the property from α except that it has a single source vertex j instead of l. Again transitive edges are omitted.

Thus β is built the following way: change all $x_{i,j}$ such that $\alpha(x_{i,j}) = 1$ to 0. Change all the symmetric values $x_{j,i}$ such that $\alpha(x_{j,i}) = 0$ to 1. These changes do not affect the value of C' since no variables from $Vars(j)$ appears in C'. After this change the variable $x_{j,l}$ to a value 1. Thus l is not a source vertex and hence $\beta(3'l)$ should be one. So we need to update the variables $y_{i,l}$ in such a way as to make $\beta(3'l) = 1$ and thereby we have $\beta(C_l) = 1$. Observe that these changes also does not affect the value of C' since no variable from $Vars(l)$ appears in C'. Thus now we have $\beta(C_I) = 1$ but $\beta(C') = 0$. Contradiction.

Thus $width(C')$ must be at least $n/6$. This completes the proof.

■

Now we are ready to prove Theorem 3.13.

Proof of Theorem 3.13 .
Consider the family of formulas $\{MDGT_n\}$. We know that $MDGT_n$ is an unsatisfiable CNF formula over n^2 variables, has constant clause width, has a polynomial size general resolution proof due to Theorem 3.14 but require resolution proof width $\Omega(n)$ due to Theorem 3.15. This completes the proof.

■

Together with Corollary 3.7, Theorem 3.13 implies an exponential separation between tree-like resolution and general resolution.

Corollary 3.19. *There is a family of formulas $\{F_n\}$ in n variables having polynomial-size resolution proofs but requiring tree-like resolution proofs of size $2^{\Omega(\sqrt{n})}$.*

Proof. Consider the family of unsatisfiable CNF formulas over n^2 variables $\{MDGT_n\}$. From Theorem 3.14, we know that $MDGT_n$ has a polynomial size general resolution refutation (i.e, $S(\vdash MDGT_n) \in n^{O(1)}$). We also have $w(MDGT_n) = O(1)$, and by Theorem 3.15, $w(\vdash MDGT_n) = \Omega(n)$. Therefore by Corollary 3.7 we have $S_T(\vdash MDGT_n) \geq 2^{w(\vdash MDGT_n) - w(MDGT_n)} = 2^{\Omega(n)}$. ■

Chapter 4

Space and Width of Resolution

We have seen two complexity measures so far: size and width. The third complexity measure of a resolution proof is space. The space of a resolution refutation is the number of clauses that have to be kept simultaneously in memory to infer a contradiction. The space used by the input clauses is not considered and these can be downloaded to the working memory when needed.

With the realization of the importance of width as a complexity measure, additional motivation was given to understand the space measure of resolution proofs, and their connection to both size and width. The investigation of resolution space was initiated by Toran in 1999 [23], and several upper and lower bounds were presented for this measure [12, 4].

In this chapter, we are following the survey by Jacobo Torán titled "Space and Width in Propositional Resolution" [22].

4.1 Definitions

Definition 4.1. *(Resolution Derivation − space oriented definition). A configuration is a set of clauses. For $k \in \mathbf{N}$, we say that an unsatisfiable CNF formula F has a space oriented resolution refutation σ bounded by clause space k if there is a sequence of configurations $F_0, F_1, \ldots, F_s$ such that $F_0 = \phi$ (i.e, starts with no clause), $F_1 \subseteq F$, $F_s = \{\square\}$ (the empty clause), in any configuration F_i there are at most k clauses and for each $i < s$, F_{i+1} is obtained from F_i by one of the following rules:*

1. **Erase** *$F_{i+1} = F_i \setminus \{C\}$ for some clause $C \in F_i$. That is, by deleting one of the clause from F_i.*

2. **Inference** *$F_{i+1} \subseteq F_i \cup \{C\}$ for C obtained by a single application of the resolution rule to two clauses $D, E \in F_i$. The clauses D, E are called the assumptions and C the conclusion of the inference step. In this step one of the clauses D or E may be erased. So F_{i+1} could also be $F_i \cup \{C\} \setminus \{D\}$.*

3. **Axiom Download** *$F_{i+1} = F_i \cup \{C\}$ for some initial clause $C \in F$. That is, by adding one of the initial clauses of F.*

For σ, the minimum k satisfying above properties is said to be the clause space of σ, denoted by $Cspace(\sigma)$. The clause space needed for the resolution of an unsatisfiable

CNF formula F, denoted by $Cspace(\vdash F)$ is the minimum k for which the formula has a space oriented refutation bounded by clause space k.

If we restrict the inference step above, such that any clause D can be used at most once as an assumption in any inference step (i.e, once used, D is erased), then we will have a tree-like space oriented resolution refutation of an unsatisfiable CNF formula F. Similarly, define the clause space needed for the tree-like resolution of F, denoted by $Cspace(\vdash_{tl} F)$, to be the minimum k for which the formula has a tree-like space oriented refutation bounded by clause space k.

If a clause C appears in F_i and F_{i+1}, we say its appearance in F_i is an **immediate predecessor** *of its appearance in F_{i+1}, and its appearance in F_{i+1} is an* **immediate successor** *of its appearance in F_i. Notice that a clause in F_{i+1} can be both an assumption, and an immediate successor of a clause from F_i. The only clause in F_{i+1} that is not an immediate successor of F_i is the conclusion of the inference. The $(i+1)^{th}$ step of the proof is the transition from F_i to F_{i+1}, and is denoted as $F_i \rightsquigarrow F_{i+1}$*

In literature, there is one more refinements of space measure:

The variable space *For an unsatisfiable CNF formula F, the variable space of F, $Vspace(\vdash F)$, is defined like the clause space of F as above, but counting the sum of widths of the clauses of the formulas F_i in the refutation, instead of just the number of clauses. More fomally, for a CNF formula F_i let us define $lit(F_i) = \sum_{C \in F_i} \{width(C)\}$. We say that F has a space oriented resolution proof σ bounded by variable space k if there is a sequence of configurations $F_0, F_1, \ldots, F_s$ such that $F_0 = \phi$, $F_1 \subseteq F$, $F_s = \{\square\}$, for any configuration F_i, $lit(F_i)$ is bounded by k and for each $i < s$, F_{i+1} is obtained from F_i by following one of the above rules. For σ, the minimum k satisfying above properties is said to be the variable space of σ, denoted by $Vspace(\sigma)$. Similar to clause space we have $Vspace(\vdash F)$ and $Vspace(\vdash_{tl} F)$.*

We have defined clause space for space oriented resolution proofs of an unsatisfiable CNF formula F. Now using the following pebbling game we will define the concept of clause space for resolution refutation of F.

Definition 4.2. *(Pebbling Game) Let $G = (V, E)$ be a connected directed acyclic graph with a unique sink s and every vertex of G has fan-in 0 or 2. The aim of the game is to put a pebble on the sink of the graph following this set of rules:*

1. *A pebble can be placed on any source vertex, that is, on a vertex with no predecessors.*

2. *A pebble can be removed from any vertex.*

3. *A pebble can be placed on an internal vertex provided both of its children are pebbled. In this case, instead of placing a new pebble on it, one can shift a pebble from a child to the vertex.*

   The minimum number of pebbles needed to pebble the unique sink following above rules is said to be the **pebbling number** *of G.*

Consider the proof graph G_π corresponding to a resolution refutation π of an unsatisfiable CNF formula F. In G_π clauses are the vertices and edges are from the

hypothesis to the conclusion of an inference rule. Clearly G_π is a DAG with initial clauses as sources and the empty clause as the unique sink. Also indegree of each vertex in G_π is either 0 or 2. Hence pebbling game is well defined in G_π. It was observed in [12] that the clause space defined for an unsatisfiable CNF formula F in Definition 4.1 coincides with the minimum number of pebbles needed for the pebble game played on the graph of a resolution refutation of F. We will present this proof in the following Lemma.

Lemma 4.3. *[12] Let F be an unsatisfiable CNF formula. $Cspace(\vdash F) = \min\{k :$ $\exists$ resolution refutation π of F, such that G_π can be pebbled with k pebbles $\}$.*

Proof. Suppose we are given a proof graph G_π corresponding to a resolution refutation π of an unsatisfiable CNF formula F. The edge directions are from hypothesis to conclusion of a resolution rule. Now given that we can play the pebble game on G_π with k pebbles, we will construct a space oriented refutation σ of F bounded by clause space k.

To begin with, in the pebbling game we have no pebbles on the vertices of G_π, correspondingly we have $F_0 = \phi$, with no clauses. Suppose we are at the i^{th} step in the pebbling game and define F_i to be the set of all vertices (clauses) where pebbles are placed. Based on the next step taken in the pebbling game we will construct F_{i+1} as follows :

1. If a source vertex (clause) is pebbled, include it in F_i to get F_{i+1}. Clearly $|F_{i+1}| \leq k$ (as we are given that the total number of pebbles placed on G_π at any point during the game is $\leq k$). This step corresponds to the step "axiom download" of Definition 4.1.

2. If a pebble is removed from a vertex, remove the corresponding clause from F_i to get F_{i+1}. Clearly $|F_{i+1}| \leq k$, as F_i has at most k clauses. This step corresponds to the step "erase" of Definition 4.1.

3. If a pebble is placed on some internal vertex C of G, then clearly by the pebbling rules both of its children are already pebbled. There are two cases: in the first case a new pebble is placed on C, in this case include clause C in F_i to get F_{i+1}. In the second case, a pebble has been shifted from one of its child to C. In this case erase that child clause and add C in F_i to get F_{i+1}. Clearly in both the cases $|F_{i+1}| \leq k$.

On the other side, suppose we are given a space oriented resolution refutation $\sigma = F_0, F_1, \ldots, F_s$ of F, bounded by clause space k (refer Definition 4.1). We will first create DAG G from σ. We show that $\exists\pi$, such that $G = G_\pi$. Finally we show that G can be pebbled with at most k pebbles.

Create a DAG G with a unique sink from σ: for each clause in σ place a vertex in G. Add edges in G corresponding to the resolution steps taken in σ. That is, for each resolution step in σ add an edge from the two hypothesis to the conclusion of the resolution rule. Clearly G is a DAG with a unique sink. Consider a sequence of vertices (say π) of G in topological order. Observe that π is a resolution refutation of F (since the end vertex in π must be the sink vertex (empty clause), and all other vertices must be a source vertex or an internal vertex with two incoming edges from previous vertices and since edges in G corresponds to resolution rule in σ the internal

vertex must correspond to a resolvent clause). This shows that $G = G_\pi$ (where G_π is a proof graph of π). Now we will show how to play the pebbling game on G with $\leq k$ pebbles.

Place pebbles in all the vertices in G_π corresponding to F_1. As they are the source vertices this is a valid step and since $|F_1| \leq k$, we have used at most k pebbles. Similarly, suppose we have placed pebbles in all the vertices corresponding to F_i. We will place pebbles to the vertices corresponding to F_{i+1} as follows:

1. If F_{i+1} is obtained from F_i by removing a clause, then remove the pebble from it. This is a valid step as a pebble can be removed from any vertex.

2. If F_{i+1} is obtained from F_i using the "inference rule" of Definition 4.1. There are two cases: in the first case we have $F_{i+1} = F_i \cup \{C\}$, where C is the conclusion and $D, E \in F_i$ are the two hypothesis of a resolution rule. Therefore, we conclude that D, E are the two children of C in G and have already been pebbled as they belongs to F_i. So put a pebble in C which is a valid step.

   In the second case, $F_{i+1} \subset F_i \cup \{C\}$, in particular, we have added clause C and removed one of the hypothesis from F_i to obtain F_{i+1}. In this case shift the pebble from the corresponding hypothesis clause in G to C.

3. If F_{i+1} is obtained from F_i by adding some initial clause C. Then put a pebble on C in G. This is a valid step as C is the source vertex in G.

Subsequently we will reach F_s and pebble the unique sink of G. At any point of time during the game we have used at most k pebbles as $\forall i, |F_i| \leq k$. ■

Thus for a resolution refutation π of an unsatisfiable CNF formula F, clause space of π, denoted $Cspace(\pi)$, is equal to the pebbling number of G_π. From Lemma 4.3, we conclude that, for F, there exists a space oriented resolution refutation σ of F bounded by clause space k iff there is a resolution refutation π of F, with the corresponding proof graph G_π which can be pebbled with at most k pebbles. We will call σ a clause space proof **conforming** to π (i.e, π and σ have the same set of clauses). We also conclude that, $Cspace(\vdash F) = k$ implies $- k$ is the minimum integer such that F has a space oriented resolution proof σ bounded by clause space k, and equivalently, k is the minimum integer such that, there is a resolution proof graph G_π which can be pebbled by k pebbles.

Now we will continue to define the concept of variable space for a resolution refutation π of an unsatisfiable CNF formula F.

Concept of Variable space of a resolution refutation of F Observe that, as each vertex of G_π corresponds to a clause of π, placing a pebble on some vertex during the pebbling game can be view as placing the corresponding clause in the memory. Thus by counting the minimum number of pebbles needed in the pebbling game played on G_π, we are actually counting the minimum number of clauses needed to be kept simultaneously in memory if we refute F via π. Similarly, instead of counting the minimum number of clauses, if we count the sum of width of clauses needed to be kept in memory, and play the pebbling game on G_π in order to minimize this sum and call such a game as Vpebbling game and the minimum sum as Vpebbling number, then following Lemma 4.3 we have the following conclusion:

Lemma 4.4. *Let π be a resolution refutation of F and G_π be the corresponding proof-graph of π. Then the Vpebbling number of G_π is k iff there is a space oriented proof σ of F conforming to π (i.e, π and σ have the same set of clauses) with variable space bounded by k.*

We will call σ a variable space proof conforming to π.

Due to Lemma 4.4, for any $\pi \vdash F$, the variable space of π, denoted $Vspace(\pi)$, is equal to the Vpebbling number of G_π. We also conclude that, $Vspace(\vdash F) = k$ implies $-$ k is the minimum integer such that F has a space oriented resolution proof σ bounded by variable space k, and equivalently, k is the minimum integer such that, there is a resolution proof graph G_π, with Vpebbling number k.

Lemma 4.5. *For any resolution refutation π of F we have $Vspace(\pi) \leq w(\pi)Cspace(\pi)$.*

Proof. From earlier discussions, we know that the pebbling number of G_π is the minimum number of clauses that has to kept simultaneously in memory in order to refute F via π. In Vpebbling game instead of counting the number of clauses we count the sum of width of clauses and since $w(\pi)$ is the maximum width among all the clauses in π, $w(\pi)$ multiplied by pebbling number of G_π is a valid output of a Vpebbling game. As Vpebbling number is the minimum over all such outputs we have, Vpebbling number of $G_\pi \leq w(\pi) \times$ pebbling number of G_π. ■

Lemma 4.6. *Any rooted binary tree of depth (length of the longest path from leaf node to the root) at most n can be pebbled with at most $n + 1$ pebbles.*

Proof. Let T be the binary tree. We will prove this by induction on the depth n of T. For base case $n = 0$, T has only one node (the unique sink) and needs only 1 pebble. For induction hypothesis, assume that any binary tree of depth less than or equal to $n - 1$ needs $\leq n$ pebbles. Now, consider a binary tree T of depth n. Let T_1 and T_2 be the two subtrees from the root. Clearly, both T_1 and T_2 have depth $\leq n-1$ and by induction hypothesis both need at most n pebbles. First pebble the subtree T_1 with at most n pebbles, leave one pebble on the root of the subtree T_1. By using one extra pebble and reusing $n-1$ pebbles from T_1, pebble the root of T_2. Thus by using at most $n + 1$ pebbles we have pebbled roots of both the subtrees, now shift any one pebble to pebble the root of T. ■

Lemma 4.7. *For an unsatisfiable CNF formula F, on n variables, $Cspace(\vdash F) \leq n + 1$.*

Proof. From Lemma 2.4, we know that every unsatisfiable CNF formula F, with n variables can be refuted by a tree-like resolution of depth at most n (by resolving at least one variable at each level). And due to Lemma 4.6, we can pebble tree-like resolution of depth at most n by $\leq n + 1$ pebbles. ■

Corollary 4.8. *For an unsatisfiable CNF formula F, on n variables, $Vspace(\vdash F) \leq 2n(n + 1)$. That is $Vspace(\vdash F) \in O(n^2)$.*

Proof. From Lemma 4.5 we know that for any $\pi \vdash F$, we have $Vspace(\pi) \leq w(\pi)Cspace(\pi)$. And since F in on n variables, $w(\pi) \leq 2n$. By Lemma 4.7 we have $Cspace(\pi) \leq n + 1$. Therefore $Vspace(\pi) \leq 2n(n + 1)$. ■

In [23] it is shown that the pigeonhole formula PHP_n^m expressing the principle that m pigeons do not fit in n holes for $m > n$, have resolution clause space exactly $n + 1$ independent of the number of pigeons.

From Corollary 3.19, we know that there are formulas with exponential gap in the size of tree-like and general refutation. Also we have seen that in case of width measure this can not happen as while transforming general resolution refutation to tree-like refutation the clauses that needs to be rederived does not increase the width of the refutation. For the case of space, it is shown in [13], that there is a family $\{F_n\}$ of formulas satisfying that F_n requires tree-like resolution clause space at least $n - 2$ but has general refutation of clause space at most $\frac{2}{3}n + 3$. Thus [13] shows that there are formulas with linear separation between the clause space measure in tree-like and general refutations. In next section we show that clause space lower bounds imply an exponential lower bounds on the size of tree-like refutation of F.

4.2 Clause Space and Tree-like Size

From Corollary 3.12, we know that width lower bounds implies size lower bounds for general resolution refutation. A natural question is: does space lower bounds can imply size lower bounds? The answer is not yet clear. However, the answer is yes for tree-like resolution refutation. That is, in [12] it is proved that for any unsatisfiable formula F, lower bounds for clause space of tree-like refutation of F imply an exponential lower bounds on the size of tree-like resolution refutation of F. We prove this in Theorem 4.11 below. For proving it we need the following definition:

Definition 4.9. *We say that a graph G_1 is embedded in a graph G_2 if a graph isomorphic to G_2 can be obtained from G_1 by adding vertices and edges or subdividing edges of G_1 (i.e, inserting vertices in the middle of edges of G_1).*

Observation 4.10. *Since any pebbling strategy for the rooted tree, also pebbles the embedded subtree it follows that the number of pebbles needed for pebbling any tree is greater than equal to the number of pebbles needed for pebbling any embedded subtree. This observation is true for any directed acyclic graph as well where pebbling game can be properly defined.*

Theorem 4.11. *[12] For an unsatisfiable CNF formula F,*

$$S_T(\vdash F) \geq 2^{Cspace(\vdash_{tl} F)} - 1$$

Proof. Let $S_T(\vdash F) = s$. Consider a tree-like proof π of F (i.e, $\pi \vdash_{tl} F$), such that $|\pi|$ is smallest. Clearly, $|\pi| = s$. Consider the proof-tree T of π. Let $d_c(T), s'$ be the depth and size respectively of the biggest complete binary tree embedded in T. We claim that the depth of biggest possible complete binary tree embedded in a tree of size s is at most $\lfloor \log_2(s+1) \rfloor - 1$ (i.e, $d_c(T) \leq \lfloor \log_2(s+1) \rfloor - 1$: as $|T| = s$, we have $s \geq s' = 2^{d_c(T)+1} - 1$. Thereby

$$\log_2(s+1) \geq d_c(T) + 1$$
$$\lfloor \log_2(s+1) \rfloor \geq d_c(T) + 1$$

Therefore we have

$$d_c(T) \leq \lfloor \log_2(s+1) \rfloor - 1 \tag{4.1}$$

We will show by induction on $|T|$ that T can be pebbled with $d_c(T)+1$ pebbles. Before presenting the proof, let us see how this proves the Theorem. We have

$$\begin{aligned} Cspace(\vdash_{tl} F) &\leq Cspace(\pi) \\ &= \text{number of pebbles needed to pebble } T, \text{ (by Lemma 4.3)} \\ &\leq d_c(T)+1 \\ &\leq \lfloor \log_2(s+1) \rfloor - 1 + 1, \text{ (from Equation 4.1)} \end{aligned}$$

which will give the desired result,

$$s \geq 2^{Cspace(\vdash_{tl} F)} - 1.$$

Now we present the proof by induction on $|T|$ that T can be pebbeled with $d_c(T)+1$ pebbles.

Base case When $|T| = 1$ we have $d_c(T) = 0$ and clearly $0+1$ pebbles are sufficient to pebble the only vertex in T.

Induction hypothesis Assume that all refutation tree T with $|T| < s$ can be pebbled with $\leq d_c(T)+1$ pebbles.

Induction step Let $|T| = s$ and T_2 be the two subtrees of the root. Then we have,

$$d_c(T) = \begin{cases} max\{d_c(T_1), d_c(T_2)\} & \text{if } d_c(T_1) \neq d_c(T_2) \\ d_c(T_1)+1 & \text{if } d_c(T_1) = d_c(T_2) \end{cases}$$

Explanation: Let $d_c(T_1) \neq d_c(T_2)$ and WLOG $d_c(T_1) < d_c(T_2)$. Then clearly $d_c(T) \not< d_c(T_2)$ as any complete binary subtree embedded in T_2 is also in T. Now if $d_c(T) > d_c(T_2)$ then there is a biggest complete binary subtree T' embedded in T and clearly T' must contains the root of T and hence it contains at least one vertex from subtrees T_1, T_2 and subsequently contains entire biggest complete binary subtree embedded in T_1, T_2. But we have $d_c(T_1) < d_c(T_2)$ hence T' is not complete. A contradiction. Thus we have $d_c(T) = d_C(T_2)$ in this case. For the second case when $d_c(T_1) = d_c(T_2)$ there must be two candidate biggest complete binary subtree T', T'' in T_1 and T_2 respectively, with depth $d_c(T_1)$ each. Consider a subtree $\tilde{T}$ containing the root node of T with T', T'' as left and right children. Clearly $\tilde{T}$ is a complete binary subtree embedding of T with depth $d_c(T_1)+1$. Also $\tilde{T}$ is the biggest such tree as T', T'' were the biggest.

By induction hypothesis one can pebble T_1 with $d_c(T_1)+1$ pebbles and T_2 with $d_c(T_2)+1$ pebbles. Let us suppose $d_c(T_1) < d_c(T_2)$, then $d_c(T) = d_c(T_2)$ as one can pebble T_2 with $d_c(T_2)+1$ pebbles, leave one pebble in the root of T_2. T_1 can be pebbled with $d_c(T_1)+1 \leq d_c(T_2)$ pebbles which we can reuse from T_2 pebbling. Thus with $d_c(T_2)+1 = d_c(T)+1$ pebbles we can pebble both the roots of T_1 and T_2, now shift any one of the two pebbles to pebble the root of T. ■

Remark 4.12. *Observe that in the proof of Theorem 4.11 we have not used the fact that π is a smallest size tree-like proof of F. Hence for any tree-like resolution proof π of F, with $|\pi| = S$, we have shown that G_π (which is a tree) can be pebbled with $d+1$ pebbles, where d is the depth of the biggest complete binary tree embedded in G_π. And since $d \leq \lfloor \log_2(S+1) \rfloor - 1$, we have $Cspace(\pi) \leq \lfloor \log_2(S+1) \rfloor \leq \log_2(S+1)$.*

Till now we have defined three complexity measures for resolution: size, width and clause space. We have seen size-width relation in Corollary 3.12 and size-clause-space relation in Theorem 4.11. The next question is whether there is some relation between width and clause space? Atserias and Dalmau [5] gave a positive answer for this problem. For a resolution proof π of an unsatisfiable CNF formula F define a complexity measure, 'internal-width' of π, denoted by $iw(\pi)$. Internal-width of π is almost similar to the width of π except that internal-width does not consider width of initial clauses of π. More formally,

$$iw(\pi) = \max_{C \in \pi \setminus F} \{width(C)\}$$

If we consider the proof graph G_π of π then $iw(\pi)$ is the width of largest clause among internal vertices of G_π. The internal width $iw(\vdash F)$ (resp. $iw(\vdash_{tl} F)$) of deriving an unsatisfiable CNF formula F in general (resp. tree-like) resolution is defined as $\min_{\pi \vdash F}\{iw(\pi)\}$ (resp. $\min_{\pi \vdash_{tl} F}\{iw(\pi)\}$).

The following observation shows relation between $w(\pi)$ and $iw(\pi)$.

Observation 4.13. *For an unsatisfiable CNF formula F and a resolution proof π of F we have*

$$w(\pi) = \begin{cases} iw(\pi) & \text{if largest clause is some } C \in \pi \setminus F \\ iw(\pi) + 1 & \text{if largest clause is some } C \in \pi \cap F \end{cases}$$

Explanation Consider the proof graph G_π of π. If the largest width clause is some internal vertex of G_π then clearly $w(\pi) = iw(\pi)$. Otherwise some source vertex C of G_π has a largest width but then the width of largest clause among internal vertices (i.e, $iw(\pi)$) is $width(C) - 1 = w(\pi) - 1$ because, due to the resolution rule and the fact that C has a largest width in G_π, the width of a conclusion clause of C is one less than $width(C)$.

Atserias and Dalmau had shown in [5] that resolution internal-width bounds the resolution clause space from below. They have defined a combinatorial characterization of resolution internal-width and used it in their proof. We are going to present this in our next section.

4.3 Combinatorial Characterization of Resolution Width

Existential k-pebble game was first introduced by Kolaitis and Vardi [17] in the context of Finite Model Theory. Atserias and Dalmau [5] provided a characterization of the resolution internal-width in terms of existential k-pebble game. In the survey [22], Jacobo Torán presented a simplified version of this game which he call Game A.

Game A is played between two players, the Spoiler and the Duplicator[1], on an unsatisfiable CNF formula F. The game is played in rounds. Both players construct a partial assignment α of the variables in F. The game starts with an empty assignment α_0 and ends with a (partial) assignment α_l, where l is the first round when partial assignment constructed so far falsifies at least one of the initial clauses of F. In each round $i, 0 \leq i < l$, of the game, Spoiler has a partial assignment α_i and can ask Duplicator for the values of a variable x in F not assigned in α_i and based on the value returned by the Duplicator, can delete some of the values assigned in $\alpha_i + x = \alpha_i'$ to construct a partial assignment α_{i+1} for the next round. Duplicator on the other hand extends α_i to α_i' by giving a boolean value for x in each round i.

The goal of Spoiler is to falsify one of the initial clauses of F with the constructed assignment while keeping the partial assignment as small as possible. Duplicator tries to keep this from happening. If the game is terminated, we define the outcome of the game to be the maximum number of variables that are assigned simultaneously in partial assignment at any moment during the game. Otherwise define the outcome to be infinity.

In particular, we may view any Spoiler strategy as a function $S(\alpha, F) \to \alpha, x$ where, S takes as input an unsatisfiable CNF formula F and a partial assignment α of variables of F and outputs a variable x in F. Similarly, any Duplicator strategy may be viewed as a function $D(\alpha, F, x) \to 0/1$ where, D takes a partial assignment α, an unsatisfiable CNF formula F and a variable x as inputs and outputs a boolean value for x. Given any Spoiler and Duplicator strategy S, D respectively, a run of a Game A played between them is defined as

$$Run_A(S, D) = \underbrace{\alpha_0, \alpha_0'}_{\text{round } 0}, \underbrace{\alpha_1, \alpha_1'}_{\text{round } 1}, \ldots, \underbrace{\alpha_l}_{\text{round } l}.$$

Let $|\alpha|$ = number of variables assigned in α. The outcome of the Game A played between S and D is defined as,

$$OC(S, D) = \begin{cases} \max\limits_{\alpha \in Run_A(S,D)} \{|\alpha|\} & \text{when game } A \text{ terminates} \\ \infty & \text{otherwise} \end{cases}$$

Consider a matrix M_A where each row corresponds to a Spoiler strategy S and each column corresponds to a Duplicator strategy D. The entries of the matrix is defined as $M_A(S, D) = OC(S, D)$. Define value of a Spoiler strategy S, denoted as $val(S)$, to be the maximum outcome of the Game A against any Duplicator strategy. That is, a maximum value in the row corresponding to S in the matrix M_A. Formally,

$$val(S) = \max_D \{OC(S, D)\}$$

Similarly, define value of a Duplicator strategy D, denoted as $val(D)$, to be the minimum outcome of the Game A against any Spoiler strategy. That is, minimum value in the column corresponding to D in the matrix M_A. Formally,

$$val(D) = \min_S \{OC(S, D)\}$$

[1]The names are justified in the context of Finite model Theory.

Definition 4.14. *For an unsatisfiable CNF formula F, define*

$$game_A(F) = \min_S \max_D OC(S, D)$$

.

Theorem 4.15. *[5] For an unsatisfiable CNF formula F,*

$$iw(\vdash F) = game_A(F) - 1$$

Proof. Let $iw(\vdash F) = k$. We will prove this in two parts. In part (i) we will show that $game_A(F) \leq k + 1$. In part (ii) we will show that $game_A(F) \geq k + 1$.

Proof of part (i) We will prove this by giving a Spoiler strategy S' such that $val(S') \leq k + 1$. This will show that

$$\exists S \; \forall D \; OC(S, D) \leq k + 1$$

Therefore we have $game_A(F) \leq k + 1$.

Spoiler strategy S': Consider proof graph G_π of a resolution refutation π of minimal internal-width for F. The strategy of S' is to follow a path P from the empty clause to one of the initial clauses in G_π maintaining the following invariant: In each round i if S' is at some clause $C \in P$ then α_i (partial assignment at the begining of round i) must falsify C (i.e, assign 0 to all the literals in C). At the $0th$ round trivially an empty assignment α_0 falsifies the empty clause in G_π. Suppose in some round i, S' is at the clause $C \in P$ and α_i falsifies C. Let C be obtained after resolving x from clauses C_1 and C_2. S' asks for the value of x. As G_π is a proof graph, any extensions of α_i to x falsifies one of the parent clauses C_1, C_2 of C. Depending on the value returned for x by the Duplicator, S' moves to the corresponding falsified clause. From the partial assignment $\alpha_i + x$ which we were calling α'_i, S' deletes the values of variables not appearing in the new clause to get a partial assignment α_{i+1} for the next round. Following this strategy S' eventually reaches an initial clause with a partial assignment falsifying it to end the game.

Observe that the number of variables assigned simultaneously by α is at most the maximal internal-width of a clause in the refutation plus the variable being resolved in each round. (note that, at the end Spoiler is on one of the initial clauses say C with a partial assignment α_l falsifying C. It may be the case that $width(C) = iw(\pi) + 1$ thereby $|\alpha_l| = iw(\pi) + 1$, however at this moment S' will no longer be asking for any further variable values since game A has already terminated).

Proof of part (ii) As $iw(\vdash F) = k$. We know that $\nexists \; \pi \vdash F$ such that $iw(\pi) = k - 1$. Using this fact we will show that there is a Duplicator strategy D' such that $val(D') > k$. Therefore we have

$$\exists D \; \forall S \; OC(S, D) \geq k + 1$$

which implies that

$$\forall S \; val(S) \geq k + 1$$

Thus $game_A(F) \geq k+1$.

Proof of existence of duplicator strategy D' with $val(D') > k$: For any Duplicator strategy D, in order to prove that $val(D) > k$ we need to prove the following statement:

Stmt Given a partial assignment α and $x \notin \alpha$ such that α does not falsify any clauses of F and $|\alpha| \leq k-1$, D must have a way to find a value $b \in \{0,1\}$ for x such that $\alpha + \{x = b\}$ does not falsify any clauses in F.

We now show that such a Duplicator strategy D' exists: D' first constructs a set of clauses $\mathcal{C}_{k-1} = \{C_1, \ldots, C_m\}$, having a resolution derivation from F of internal-width at most $k-1$. As $\nexists\ \pi \vdash F$ such that $iw(\pi) = k-1$ we know that $\square \notin \mathcal{C}_{k-1}$. As internal-width does not consider the width of initial clauses it follows that $F \in \mathcal{C}_{k-1}$. We show that there is a Duplicator strategy D' satisfying a stronger statement $Stmt1$:

Stmt1 Given a partial assignment α and $x \notin \alpha$ such that α does not falsify any clauses of $\mathcal{C}_{k-1}$ and $|\alpha| \leq k-1$, then D' has a way to find a value $b \in \{0,1\}$ for x such that $\alpha + \{x = b\}$ does not falsify any clauses in $\mathcal{C}_{k-1}$.

Suppose not. Then let $\alpha + \{x = 0\}$ falsifies a clause $C_i \in \mathcal{C}_{k-1}$. This implies that $C_i = C_i' \cup \{x\}$ (because previously it must be the case that α leaves C_i undicided, i.e, some literals of C_i are not assigned any values in α however after assigning x a value 0, we have $\alpha(C_i) = 0$). This implies that $\alpha(C_i') = 0$. As $|\alpha| \leq k-1$ we have $width(C_i') \leq k-1$.

Similarly let $\alpha + \{x = 1\}$ falsifies a clause $C_j \in \mathcal{C}_{k-1}$. By the similar argument, this implies that $C_j = C_j' \cup \{\neg x\}$. Therefore we have $\alpha(C_j') = 0$ and since $|\alpha| \leq k-1$ we have $width(C_j') \leq k-1$.

Combining the two, we have $\alpha(C_i' \vee C_j') = 0$ and as $|\alpha| \leq k-1$ we have $width(C_i' \vee C_j') \leq k-1$. Observe that $C_i' \vee C_j'$ can be obtained from C_i and C_j by resolving variable x. That is, $\frac{C_i \quad C_j}{C_i' \vee C_j'}$. And since internal-width does not consider the width of initial clauses $C_i' \vee C_j' \in \mathcal{C}_{k-1}$. This is a contradiction since it shows that α was already falsifying a clause in $\mathcal{C}_{k-1}$.

■

4.4 Width vs Space

Now we are ready to present internal-width and clause-space relation of resolution proved by Atserias and Dalmau in 2003 [5].

Theorem 4.16. *[5] For an unsatisfiable CNF formula F,*

$$Cspace(\vdash F) \geq iw(\vdash F) - w(F) + 1$$

Proof. Let $w(F) = w$. There are two cases:

Case (i) $iw(\vdash F) < w$. That is, $iw(\vdash F) \leq w - 1$. This implies that $iw(\vdash F) - w + 1 \leq 0$. Thus we need to prove that

$$Cspace(\vdash F) \geq 0 \geq iw(\vdash F) - w(F) + 1$$

which is trivially true. Since by definition $Cspace(\vdash F) \geq 0$.

Case (ii) $iw(\vdash F) \geq w$. Therfore $iw(\vdash F) = k + w$ for some k. We will show by the way of contradiction that $Cspace(\vdash F) \geq k + 1$, which will prove the theorem.

Suppose not. Then following Definition 4.1, there is a space oriented resolution refutation $\pi = F_0, F_1, \ldots, F_m$ bounded by clause space k. That is, π satisfies the following rules: $F_0 = \phi, F_1 \subseteq F$, $F_m = \{\square\}$ (the empty clause), in each F_i there are at most k clauses and for each $i < m$, F_{i+1} is obtained from F_i by following rules $1, 2$ and 3 of Definition 4.1.

We will show that $\forall i, 1 \leq i \leq m$, F_i is satisfiable. This will imply that F_m is also satisfiable, which means $\square \notin F_m$. Contradiction.

We will assume WLOG that F_1 has only one clause, because if say F_1 has l clauses, $C_1, \ldots, C_l$, then we can convert the refutation π to refutation $\pi' = F_0, F_{11} = \{C_1\}, F_{12} = \{C_2\}, \ldots, F_{1l} = \{C_l\}, F_2, \ldots, F_m$ of F bounded by clause space k.

Let s_i be the number of clauses in F_i. Clearly $\forall i, s_i \leq k$ by assumption. We will give a Spoiler strategy S'' which will play on F such that $\forall D$, when S'' plays against D, the following holds: at each round i, S'' has either a partial assignment α_i, with $|\alpha_i| \leq s_i \leq k$ and α_i satisfies F_i or S'' has an α_i, with $|\alpha_i| \leq k + w$ and the game A terminates, i.e, at least one of the initial clauses of F gets falsifies by α_i. Before describing the Spoiler strategy S'', let us see how this will give us a contradiction.

We have $iw(\vdash F) = k + w$. Recall from the previous section that there exists a Duplicator strategy D such that against any Spoiler strategy S, $OC(D, S) > k + w$. That is

$$\exists D \; \forall S \; OC(S, D) > k + w$$

Fix such a duplicator strategy D. Now consider $Run_A(S'', D)$. Suppose the game A terminates at some round i. Therefore we have $|\alpha_i| \leq k + w$ and $\forall j < i$, $|\alpha_j| \leq k$. Therefore $OC(S'', D) \leq k + w$. Contradiction.

Therefore the game A does not terminate. Which implies that α_m is defined such that $|\alpha_m| \leq k$ and α_m satisfies F_m. The required Contradiction.

Spoiler Strategy S'' : We will give the strategy by induction on round i. We describe at one shot what S'' does in many rounds. All intermediate rounds satisfies size requirement.

Base case: When $i = 1$ we have $F_1 = C$ and $width(C) \leq w$, since $w(F) = w$. Initially α_0 is an empty assignment. S'' asks Duplicator the value of each of the literals of C. If Duplicator returns a value 1 for any one of these literals, S'' just set that literal 1 in α_0 to get α_1, with $|\alpha_1| = 1 = |F_1|$ and clearly α_1 satisfies F_1. Otherwise, Duplicator returns a value zero for all the literals and S'' sets all literals zero in α_0 to get α_1, with $|\alpha_1| \leq w \leq k + w$, and α_1 falsifies the initial clause C and the game terminates.

Induction step: Let we have α_{i-1}, with $|\alpha_{i-1}| \leq s_{i-1}$ and α_{i-1} satisfies all the clauses of F_{i-1} and still the Game A has not terminated. That is no initial clauses are falsified yet. Now we have to construct α_i such that $|\alpha_i| \leq s_i$ and α_i satisfies F_i, or $|\alpha_i| \leq k + w$ and the Game A terminates. Based on how F_i is obtained from F_{i-1} we have three cases (see Definition 4.1):

Case 1: Erase Let F_i is obtained from F_{i-1} after deleting a clause C_l. In α_{i-1}, if literal l_l is responsible to satisfy C_l and for all other clauses in F_{i-1}, we have different literals responsible for satisfying them, then α_i can be obtained from α_{i-1} just by deleting l_l from α_{i-1}. Clearly α_i satisfies F_i and also we have,

$$|\alpha_i| + 1 = |\alpha_{i-1}| \leq s_{i-1} = s_i + 1$$

which implies $|\alpha_i| \leq s_i$ as required. In other case when l_l is the only literal responsible to satisfy C_l and to some other clause C_i then we know that $\alpha_{i-1} < s_{i-1}$. We will set $\alpha_i = \alpha_{i-1}$. Clearly α_i satisfies F_i and,

$$|\alpha_i| = |\alpha_{i-1}| < s_{i-1} = s_i + 1$$

which implies $|\alpha_i| \leq s_i$ as required.

Case 2: Inference Let F_i is obtained from F_{i-1} by adding the resolvent of some two clauses of F_{i-1}. In this case S'' sets $\alpha_i = \alpha_{i-1}$. This is because α_{i-1} itself satifies F_i as it satisfies the two parent clauses in F_{i-1} hence it will also satisfy the resolvent clause in F_i and clearly we have,

$$|\alpha_i| = |\alpha_{i-1}| \leq s_{i-1} \leq s_i$$

which implies $|\alpha_i| \leq s_i$ as required. In the above Equation the last inequality is not strict because inference rule may delete one of the parent clause as well.

Case 3: Axiom Download Let F_i is obtained from F_{i-1} by adding an initial clause C. In this case S'' asks Duplicator for the values of all the literals in C that are not assigned at this point. If any one of the literals is set to 1 by the Duplicator then S'' extends α_{i-1} to α_i by setting the corresponding literal to 1. In which case clearly α_i satisfies F_i and we have,

$$|\alpha_i| = |\alpha_{i-1}| + 1 \leq s_{i-1} + 1 = s_i$$

which implies $|\alpha_i| \leq s_i$ as required. In other case when Duplicator returns a 0 for all the literals in C, S'' extends α_{i-1} to α_i by assigning 0 to all the literals of C. As there are at most w literals in C we have,

$$|\alpha_i| \leq |\alpha_{i-1}| + w \leq s_{i-1} + w \leq k + w$$

which implies $|\alpha_i| \leq k + w$, and in this case α_i falsifies the initial clause C and the game A gets terminated.

■

We end this chapter by giving a combinatorial characterization of tree-like resolution space.

4.5 Combinatorial Characterization of Tree-like Resolution Clause Space

Impagliazzo and Pudlák in [20] gave a combinatorial characterization for tree-like resolution clause space in terms of a 2-player game. Torán in [22] called this game as Game B.

Game B is played between two players, the Prover and the Delayer, on an unsatisfiable CNF formula F. The game is played in rounds. Both players construct in steps partial assignment α for the formula. The game starts with an empty assignment α_0 and ends with a partial assignment α_l where, l is the first round when at least one of the initial clauses of F gets falsified by the partial assignment constructed so far. In each round $i, 0 \leq i < l$, Prover has a partial assignment α_i and chooses a variable x to be assigned that has not yet assigned. Then Delayer chooses one of $0, 1$, or $*$ for the variable x in the same round. If 0 or 1 is chosen, no points are scored by the Delayer and the variable is set to the chosen bit and the next round begins with the updated partial assignment α_{i+1}. If $*$ is chosen, then Delayer scores one point, but Prover then can choose the value for the variable to construct an updated partial assignment α_{i+1} for the next round. When a variable has been assigned a value, it remains with this value until the end of the game.

The goal of Delayer is to score as many points as possible and Prover tries to prevent this. The outcome of the game is the number of points scored by Delayer.

In particular, we may view any Prover strategy as a function $P(\alpha, F) \to \alpha, x, b$, where x to be queried, and assigned b if D says $*$. Similarly, any Delayer strategy may be view as a function $D(\alpha, F, x) \to 0/1/*$. For any Prover and Delayer strategies P, D respectively, a run of Game B played between them is defined as:

$$Run_B(P, D) = \alpha_0, \alpha_1, \ldots, \alpha_l.$$

The outcome of the Game B played between P and D is defined as:

$$OC(P, D) = \text{Number of times Delayer returns } * \text{ in the } Run_B(P, D)$$

Consider a matrix M_B where each row corresponds to a Prover strategy P and each column corresponds to a Delayer strategy D. The entries of the matrix is defined as $M_B(P, D) = OC(P, D)$. Define value of a Prover strategy P, denoted as $val(P)$, to be the maximum outcome of the Game B played against any Delayer strategy. That is, a maximum entry in the row corresponding to P in the matrix M_B. Formally,

$$val(P) = \max_D OC(P, D)$$

Call a Prover strategy with minimum value as an optimal Prover strategy, denoted as P^*. Thus for P^* we have,

$$val(P^*) \leq val(P), \forall P$$

Similarly, define value of a Delayer strategy D, denoted as $val(D)$, as the minimum outcome of the Game B played against any Prover strategy. That is, a minimum

entry in the column corresponding to D in the matrix M_B. Formally,

$$val(D) = \min_P OC(P, D)$$

Call a Delayer strategy with maximum value as an optimal Delayer strategy, denoted as D^*. Thus for D^* we have,

$$val(D^*) \geq val(D), \forall D$$

Definition 4.17. *For an unsatisfiable CNF formula F, define*

$$game_B(F) = \max_D \min_P OC(P, D) = \max_D val(D) = val(D^*).$$

Theorem 4.18. *[13] For an unsatisfiable CNF formula F*

$$Cspace(\vdash_{tl} F) = game_B(F) + 1$$

Proof. Let $Cspace(\vdash_{tl} F) = s$. We will prove the theorem in two parts: (i) $game_B(F) \geq s - 1$ and (ii) $game_B(F) \leq s - 1$.

Proof of part (i) For proving this we will give a Delayer strategy D' such that $val(D') \geq s-1$. That is, D' scores at least $s-1$ points against any Prover strategy. This will give the desired result as,

$$game_B(F) = val(D^*) \geq val(D') \geq s - 1$$

Delayer strategy D' : We will give the Delayer strategy based on induction on the number of variables in F. Let n denotes the number of variables in F.

Base Case: When $n = 1$, F has only one variable and hence $F = x \wedge \neg x$. Clearly tree-like resolution refutation for F contains only two leaf nodes $x, \neg x$ and a root node for the empty clause (i.e, $\square$). Therefore $s = 2$ in this case. D' just returns a $*$ for the value of x asked by the Prover and scores 1 points.

Induction Step: For $n > 1$, let x be the first variable asked by Prover and let $F|_{x=1}$ and $F|_{x=0}$ be the CNF formulas obtained after assigning 1 and 0 respectively to variable x in F. Clearly in $F|_{x=1}$ (resp. $F|_{x=0}$) all the clauses containing x (resp. $\neg x$) becomes 1 and $\neg x$ (resp. x) has been dropped from all the clauses containing it. Delayer D' constructs an optimal clause space tree-like resolution refutation T_1 for the formula $F|_{x=1}$. Let s_1 represents the clause space of T_1. Put back $\neg x$ in T_1, that is put back $\neg x$ in all the clauses of $F|_{x=1}$ from where it has been dropped and propagate it up in T_1 through the resolution rule. Call the tree obtained as T_1'. See Figure 6(a). There are two possibilities: either the root of T_1' contains $\neg x$ or $\square$. In the second case clearly $s_1 = s$ and existence of T_1' shows that F can be refuted just from the clauses of F that do not contain x. Thus D' returns 1 as the value of x. By the induction hypothesis, D' can score at least $s_1 - 1 = s - 1$ points playing the game on $F|_{x=1}$.

In the first case, i.e, when the root of T_1' contains $\neg x$, D' constructs an optimal clause space tree-like resolution refutation T_2 for the formula $F|_{x=0}$. Let s_2 represents

the clause space of T_2. Similarly put back x in T_2 to get T_2'. See Figure 6(b). If the root of T_2' contains the empty clause $\Box$ then $s_2 = s$. In this case D' returns a 0 as the value of x. By induction hypothesis, D' can score at least $s_2 - 1 = s - 1$ points playing on $F|_{x=0}$.

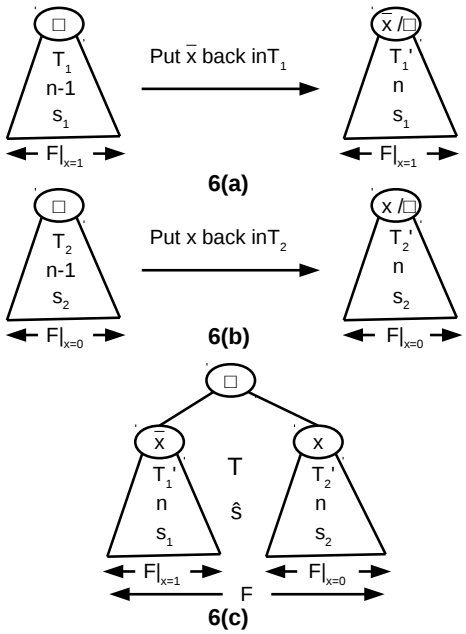

Figure 6: (a) D' constructs minimum clause space tree-like refutation for $F|_{x=1}$. Then put back $\bar{x}$ in T_1 to get T_1'. If empty clause appears at the top, D' returns x=1 and plays in T_1'. Otherwise, (b) D' constructs T_2' from $F|_{x=0}$ as in (a). If empty clause appears in top of T_2', D' returns x=0 and plays in T_2'. (c) If $\bar{x}$, x appears at the top of T_1', T_2' respectivley, D' construct T as above and plays in that subtree which has maximum clause space.

The remaining case is when the roots of T_1', T_2' contains $\neg x, x$ respectively. In this case, D' construct a tree-like resolution refutation T of F using T_1', T_2'. To construct T, D' resolved $\neg x, x$ at the roots of T_1', T_2' to get an empty clause ($\Box$) at the top of T. Let $\hat{s}$ represents the clause space of T. See Figure 6(c). Clearly we have $s \leq \hat{s}$. We have two subcases:

Subcase 1 ($s_1 = s_2$) In this case we have $\hat{s} = s_1 + 1$ (first pebble the subtree T_1' using s_1 pebbles, leave one pebble on the root of T_1'. By using one extra pebble and reusing $s_1 - 1$ pebbles from T_1', pebble T_2' which require $s_2 = s_1 - 1 + 1$ pebbles. Thus with total s_1+1 pebbles we pebbled roots of both the subtrees of T. Reuse one of them to pebble its root). This implies that $s_1 = \hat{s} - 1 \geq s - 1$. Similarly, $s_2 \geq s - 1$.

In this case, D' answers a $*$ as the value of x asked by the Prover and scores one point. Again by induction hypothesis, D' scores at least $s - 2$ more points by playing the game in either of the formulas $F|_{x=1}$ or $F|_{x=0}$.

Subcase 2 ($s_1 \neq s_2$) In this case $\hat{s} = max\{s_1, s_2\}$ (let $s_1 < s_2$, then pebble T_2' with s_2 pebbles. Leave one pebble on the root of T_2', and by reusing $s_1 \leq s_2 - 1$

pebbles, pebble T'_1. Thus by using only s_2 pebbles both the children of the root of T have been pebbled, shift one of them to pebble the root. s_2 is the minimum number of pebbles to pebble T because we are given that this is the minimum to pebble T'_2). This implies that $max\{s_1, s_2\} \geq s$, as $\hat{s} \geq s$.

In this case, D' returns the value leading to the formula that requires tree-like resolution space at least s and thereby can score at least $s-1$ points by induction hypothesis playing on the corresponding formula. For example, D' returns $x = 1$, if $s_1 > s_2$ and starts playing on the formula $F|_{x=1}$, which has $n-1$ variables and tree-like resolution clause space at least s hence by induction hypothesis D' scores at least $s-1$ points.

Proof of part (ii) We need to prove: $game_B(F) = val(D^*) \leq s-1$. To prove this we will first define a partial Prover strategy P' and then based on the Game B played between D^* and P' we will prove the desired result.

Partial Prover strategy P' : Consider a tree graph T_π of a tree-like resolution refutation π of F with minimum clause space. The partial Prover strategy P' is to follow a path Q from the empty clause to one of the initial clauses in T_π maintaining the following invariants: in each round i, if P' is in some clause $C \in Q$ then α_i (partial assignment constructed during the Game B in round i) must falsifies C (i.e, assigns 0 to all the literals in C) and if Delayer returns a $*$ as the value of variable x queried in round i then P' has to mark C with $*$.

In round 0, P' has an empty assignment α_0 which trivially falsifies the empty clause. Suppose in T_π variable x is resolved to get an empty clause. P' asks Delayer for the value of x. If Delayer returns a 0/1 for the value of x, P' updates the partial assignment to α_1 accordingly and goes to that parent clause of $\square$ which gets falsified by α_1 (by the nature of resolution one of the parent has to get falsified). If the Delayer returns a $*$ then P' marks the empty clause with a $*$ and chooses any value 0/1 as the value of x, updates the partial assignment to α_1 and goes to that parent clause of $\square$ which gets falisfied by the updated assignment α_1.

In particular, in round i, P' is in some clause $C \in Q$ which is falsified by the partial assignment α_i. Let C_0 and C_1 be the parent clause of C and C is obtained by resolving variable x. P' asks Delayer for the value of x. If Delayer returns 0/1, P' updates α_i to α_{i+1} accordingly and goes to a parent clause C_0 or C_1 which gets falsified by α_{i+1}. If Delayer returns a $*$, P' marks C with a $*$, chooses a value 0/1 for x, updates α_i to α_{i+1} accordingly, and moves to C_0 or C_1 which gets falsified by α_{i+1}.

Clearly P' is a partial strategy. Consider the set of all Prover strategies extending P'. That is,

$$\mathcal{P} = \{P'' | P'' \text{extends } P'\}$$

Fix a Delayer strategy D^* (recall this is an optimal Delayer strategy). Let us denote $T(D^*, \pi)$ the subtree of T_π formed by all the clauses that can be visited by P' and the edges joining them in the Game B played between D^* and P'. Subtree $T(D^*, \pi)$ has the following properties:

1. It contains the empty clause $\square$ (as P' starts Game B from the empty clause of T_π).

2. It contains all the $*$ marked nodes of T_π and they are the only branching nodes of $T(D^*, \pi)$ (as from marked clause P' can move to any one of its parent clause).

3. The pebbling number (recall Definition 4.2) of $T(D^*, \pi)$ is less than equal to the pebbling number of T_π (indeed this is true for any subgraph since any pebbling strategy for a graph works for its subgraph as well).

4. For each path Q from the empty clause to a initial clause in $T(D^*, \pi)$, there is a Prover strategy $P \in \mathcal{P}$ such that Q represents a run of the Game B played between P and D^*.

5. As $val(D^*) = game_B(F)$, every path from an empty clause to some initial clause in $T(D^*, \pi)$ contains at least $game_B(F)$ number of $*$ marked nodes.

Thus we have,

$$\begin{aligned} Cspace(\vdash_{tl} F) &= s \\ &= \text{minimum number of pebbles required to pebble } T_\pi \\ &\geq \text{minimum number of pebbles required to pebble } T(D^*, \pi) \\ &\geq game_B(F) + 1, \quad \text{From Lemma 4.19 (below)} \end{aligned}$$

This will prove part (ii), that is, $game_B(F) \leq s - 1$.
So put together part (i) and (ii), we have shown that $game_B(F) = s - 1$, which completes the proof of Theorem 4.18.

■

Lemma 4.19. *Minimum number of pebbles needed to pebble $T(D^*, \pi)$ is at least $game_B(F) + 1$.*

Proof. Consider any optimal pebbling strategy for $T(D^*, \pi)$. We will show that this will take at least $game_B(F) + 1$ pebbles.

At the end of any pebbling game, all paths from the empty clause (root) to any initial clause (leaf) in $T(D^*, \pi)$ has a pebble because root has a pebble and it lies in all such paths. Let m be the first moment when all root-to-leaf paths in $T(D^*, \pi)$ have a pebble. At $(m-1)$, some root-to-leaf path say ρ does not have a pebble. Clearly at moment m we must place a pebble at the leaf of ρ which is an initial clause say C. This is because by the pebbling rules, we can not place pebbles on any internal node of ρ as all of its children are not already pebbled. And once that pebble is placed on C, all root-to-leaf paths have a pebble, so before placing that pebble, all such paths except ρ already have a pebble.

Now consider all the paths branching out from ρ. All of these already have a pebble at time m-1. But the pebble cannot be in the part of the path common with ρ. So each part hanging out has a pebble. There are at least $game_B(F)$ paths hagging out (from properties 2 and 5 of $T(D^*, \pi)$), all disjoint. So we have at least $game_B(F)$ pebbles at moment $m-1$ and one more at moment m. ■

Chapter 5

Size-Width Tradeoffs for Tree-like Resolution

In [7] Eli Ben-Sasson investigates tradeoffs of various basic complexity measures such as size, space, and width. He showed examples of formulas that have optimal resolution proofs with respect to any one of these parameters, but optimizing one parameter must cost an increase in the other. However, in this chapter we will present the basic tradeoff result for only tree-like resolutions from [7]. The tradeoffs is exposed by a family of pebbling contradictions.

Definition 5.1. *(Pebbling Contradictions) [7]. Let $G = (V, E)$ be a directed acyclic graph in which every vertex has fan-in 2 or 0 with a unique sink s. We call a graph with these properties a circuit-graph. Associate a boolean variable x_v with every vertex $v \in V(G)$. Peb_G, the pebbling contradiction of G is the conjunction of:*

(1) **Source axioms** *x_v for each source v.*

(2) **Sink axiom** *$\neg x_s$ for unique sink s of G.*

(3) **Pebbling axioms** *$\neg x_u \vee \neg x_w \vee x_v$, for u, w the two predecessor of v and for each internal vertex v of G.*

Observation 5.2. *Peb_G is an unsatisfiable 3-CNF formula.*

Proof. To make an assignment α which evaluates Peb_G to 1, we have to set all clauses of type (1) 1 in α, thereby variables corresponding to all internal vertices have to be set to 1 including the variable for s, (i.e, x_s) , hence clause of Type (2) can not be satisfied. On the other hand, if we start constructing a satisfying assignment by setting $\neg x_s = 1$, then due to the clause of type (3) for s, variables corresponding to one of its children has to be assigned 0, thereby one of the variables for the source node has to be assigned a 0, hence in this case not all the clauses of type (1) can be satisfied. ■

Also observe that Peb_G is a Horn CNF formula with $|V|$ variables and $|V| + 1$ clauses.

Eli Ben-Sasson shows that for certain circuit-graphs G with n vertices, the tree-like resolution of the formulas Peb_G presents the following tradeoff:

Theorem 5.3. *[7] For infinitely many integers n, there exist contradictions F_n over n variables of size (number of clauses) $n+1$ such that:*

1. *F_n is a Horn CNF (i.e, every clause has at most one positive literal).*
2. *$S_T(\vdash F_n) = O(n)$ and $Cspace(\vdash F_n) = O(1)$. Moreover, there exist tree-like proofs of F_n with linear size and constant clause space simultaneously.*
3. *$w(\vdash F_n) = O(1)$.*
4. *For any tree-like proof π of F_n:*

$$w(\pi) \cdot \log_2 |\pi| \geq w(\pi) \cdot Cspace(\pi) \geq \Omega\Big(\frac{n}{\log_2 n}\Big).$$

As already pointed out, the tradeoff will be exposed by the pebbling contradiction. Given any circuit-graph $G = (V, E)$, we first show that Peb_G has a linear size and with width $O(1)$ resolution proof. This in particular proves part 3 of Theorem 5.3.

Lemma 5.4. *[7] For any circuit-graph $G = (V, E)$, there exists a resolution proof π of Peb_G such that $|\pi| = O(|V|)$ and $w(\pi) = O(1)$.*

Proof. **Resolution proof π is as follows:**

Fix a topological ordering on the vertices V of G. In this order we derive inductively for each $v \in V$ the clause (x_v).

Base case In the topological ordering all source vertices of G appears at the beginning. The variables corresponding to them are precisely the initial clauses of type (1). Hence they come for free.

Induction step Suppose we are at some internal vertex $v \in V$. By induction hypothesis variables (clauses) corresponding to all the vertices before v in the topological odering have already been derived. Let u and w be the two children of v in G. Using the initial clause of type (3) for v and the clauses x_u, x_w we derive x_v as follows:

$$\dfrac{\dfrac{(x_u) \quad (\neg x_u \vee \neg x_w \vee x_v)}{x_w}\, x_u \quad (x_w) \quad (\neg x_w \vee x_v)}{x_v}\, x_w$$

Thus at last we will derive the clause x_s and using the initial clause of type (2) (i.e, $\neg x_s$) we have a contradiction.

Size and width of π Each internal clauses is derived using two resolution steps and we use one more step to derive the empty clause at the end. Thus we have $|\pi| \leq 2|V| + 1 = O(|V|)$. Clearly every clause $C \in \pi$ has $width(C) \leq 3$. Thus $w(\pi) = O(1)$. ■

In order to prove part 2 of Theorem 5.3, we need the concept of decision trees.

Decision trees
Let F be a CNF formula over n variables and m clauses. A **search problem** for F is the following: given an assignment α of the variables of F, find a clause $C_i \in F, 1 \leq i \leq m$ such that $\alpha(C_i) = 0$, if there is such a clause, otherwise answer 1.

Definition 5.5. *(Decision trees for CNF search problems) A decision tree is a binary tree, with internal vertices labeled by variables, edges labeled by* 0 *or* 1*, and leaves labeled with the possible outputs. Every assignments to the variables defines a path through the tree in the natural way, and the label at the end of the path is said to be the output of the decision tree on that assignment.*

We say that D is a decision tree for the search problem for F if D correctly solves F on every input (assignment). For a CNF formula F, let $S_D(F)$ denote the minimal size of a decision tree solving the CNF search problem for F.

The following Lemma shows that decision trees for an unsatisfiable CNF formula F are closely related to tree-like resolution proof of F:

Lemma 5.6. *[8] For an unsatisfiable CNF formula F, $S_T(\vdash F) = S_D(F)$.*

Proof. The tree of the resolution refutation is a decision tree, where each internal vertex is labeled by the variable resolved upon at that step. Hence $S_T(\vdash F) \geq S_D(F)$. More formally, let T be a smallest size tree-like resolution proof of F. We will show by induction on $|T|$ that there is a decision tree with the same tree structure as T which solves the search problem for F.

Base case When $|T| = 1$, T has just an empty clause. Thus F must contain an empty clause. Corresponding decision tree consist of a single vertex labeled by the index corresponding to the empty clause of F.

Induction step When $|T| > 1$, let x be the last variable on which the refutation resolves. Let T_0, T_1 be the two subtrees from the root inferring say x and $\neg x$ respectively. From Lemma 3.8, for $b \in \{0, 1\}$ the tree-like resolution T_b can be restricted to a refutation of $F|_{x=b}$. By induction hypothesis, T_b can be transformed into a decision tree D_b which solves the search problem for $F|_{x=b}$. Clearly, the search problem for F can be solved by a decision tree which queries x and if $x = b$ it applies the decision tree D_b.

For the opposite direction (i.e, for proving $S_D(F) \geq S_T(\vdash F)$), we claim that given a decision tree D, we can derive from it a tree-like refutation without increasing its size. As F is unsatisfiable, every leaf of D is labeled by a clause, since 1 is not a legitimate answer.

Look at the two leaves labeled C_i, C_j with their parent v labeled x. If x does not occur in one of the two clause (WLOG. say C_i), then label v with C_i, erase its sons and make the tree smaller. Otherwise, WLOG. x must appear in C_i in positive form and in C_j in negative form. In this case we label v with the consequence

of resolving C_i, C_j on x. Continuing in this way up through the decision tree we conclude $S_D(F) \geq S_T(\vdash F)$. ■

Now the following Lemma proves part 2 of Theorem 5.3.

Lemma 5.7. *[7] For any circuit-graph $G = (V, E)$, there exists a tree-like resolution proof π of Peb_G with $|\pi| = O(|V|)$ and $Cspace(\pi) = O(1)$.*

Proof. For the searching problem of unsatisfiable CNF formula Peb_G, we will construct a decision tree D of linear size which can be pebbled with constant number of pebbles. Thereby using Lemma 5.6 we have a tree-like resolution refutation T of linear size and constant clause space. We will use the term vertex to denote the vertices of G and nodes to denote the vertices of the decision tree D.

Linear size decision tree for Peb_G Define a topological ordering of the vertices of G, and query vertices according to this order. If one of the answer (0 or 1) leads to a falsifying assignment to some initial clause, label this answer with that clause and proceed via the other answer. If both the answers lead to a falsifying assignment to some clauses then label both the answers with the corresponding clauses and stop.

Claim 5.8. *When a vertex v is queried, answering 0 leads to a falsifying assignment.*

Proof. We will prove this by induction on the topological ordering on the vertices of G. If v is a source vertex, then answering 0 will falsifies the source axiom x_v. So let v is any internal vertex with children u, w. Since we are quering vertices in topological ordering, we must have queried vertices u, w. By induction hypothesis, we must have answered both queries by 1, because answering 0 would lead to a falsifying assignment. So we have $u = w = 1$ and answering $v = 0$ will falsifies the pebbling axiom $(\neg x_u \vee \neg x_w \vee x_v)$. This completes the proof of Claim 5.8. ■

By Claim 5.8, each node in D has one branch that leads immediately to a falsifying assignment. At last when we query the sink s, answering 1 falsifies the sink axiom. Thus for s both answer falsifies some clauses which completes the decision tree. Thus the resulting decision tree D is just a linear path of n nodes with a subtree of size one hanging at every node. Thereby D can be pebbled with constant number of pebbles. ■

We have seen that part $(1), (2)$ and (3) of Theorem 5.3 is true for Peb_G, constructed from any circuit graph G. We now give an intuition, in Remark 5.9 (below) that this is not true for part (4) of Theorem 5.3. That is, part (4) of Theorem 5.3 is not true for Peb_G, constructed from any circuit graph G.

Remark 5.9. *Let $G = (V, E)$ be any circuit graph, with a unique sink s. For similicity let G be as in Figure 7. Inspired from the above proof, one may derive a contradiction of Peb_G as follows: consider vertices of G in reverse topological order: $s, w, v, q, t, p, a, b, c, d, e$. Perform the resolution steps based on this order and call the conclusion of i^{th} resolution step as clause (i).*

Resolution step (1) *Apply resolution rule on the sink axiom and pebbling axiom corresponding to s, and resolve variable x_s.*

$$\frac{(\neg x_s) \quad (\neg x_w \vee \neg x_v \vee x_s)}{(\neg x_w \vee \neg x_v)}\, x_s$$

View clause (1) as a cut in G, which separates vertex s from rest of the graphs (see Figure 7).

Resolution step (2) *Apply resolution rule on clause (1) and pebbling axiom corresponding to internal vertex w, and resolve variable x_w.*

$$\frac{(\neg x_w \vee \neg x_v) \quad (\neg x_q \vee \neg x_t \vee x_w)}{(\neg x_v \vee \neg x_q \vee \neg x_t)}\, x_w$$

View clause (2) as a cut in G, which separates s and w from the rest of G (see Figure 7). Similarly, we have clause (3) to (6). Clearly clause (6) $= (\neg x_a \vee \neg x_b \vee \neg x_c \vee \neg x_d \vee \neg x_e)$ separates all internal vertices from the source vertices of G.

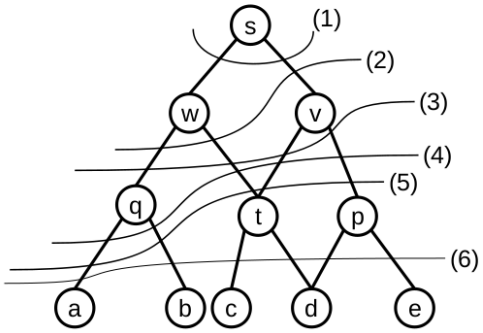

Figure 7: G = (V, E), a circuit graph, with a unique sink s. Assume edges are directed towards s. Clauses (i) which results as the conclusion of the i^{th} resolution step in the resolution proof of Peb_G may be viewed as a cut in G.

Now by applying resolution rule on clause (6) and the source axioms, we get an empty clause:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{(x_a) \quad (\neg x_a \vee \neg x_b \vee \neg x_c \vee \neg x_d \vee \neg x_e)}{(x_b) \quad (\neg x_b \vee \neg x_c \vee \neg x_d \vee \neg x_e)}\, x_a}{(x_c) \quad (\neg x_c \vee \neg x_d \vee \neg x_e)}\, x_b}{(x_d) \quad (\neg x_d \vee \neg x_e)}\, x_c}{(x_e) \quad (\neg x_e)}\, x_d}{\square}\, x_e$$

Size and clause space of the proof *Clearly the proof mentioned is a tree-like resolution proof. The proof is a linear path of n vertices with a subtree of size one hanging at every vertex. Hence the proof has linear size and constant clause space.*

Now consider the width of the proof. As seen in Figure 7, the width of this proof is actually a largest cut in G which separates s from the source vertices. Thus the conclusion of these remark is, in order to prove part 4 of Theorem 5.3, one should consider circuit-graphs which does not contain any cuts of smaller size separating sink vertex from source vertices.

We will now prove that part (4) of Theorem 5.3 holds for Peb_G constructed from any circuit-graph G on n vertices which has large black-white pebbling number, denoted $BW\text{-}Peb(G)$ (defined below in Definition 5.11). To be precise, consider any graph $\tilde{G}_n$ on n vertices with $BW\text{-}Peb(\tilde{G}_n) = \Omega(n/\log n)$. The existence of such a graph has already been proved by Gilbert and Tarjan in 1978 [15]. Here we are stating their theorem without proof:

Theorem 5.10. *[15] For arbitrarily large n, there exists circuit-graphs $\tilde{G}_n$ on n variables such that*

$$BW\text{-}Peb(\tilde{G}_n) = \Omega\Big(\frac{n}{\log n}\Big)$$

Now for the circuit graph $\tilde{G}_n$, define an unsatisfiable CNF formula $Peb_{\tilde{G}_n}$. We will prove that part 4 of Theorem 5.3 holds for $Peb_{\tilde{G}_n}$. Before proving this we need to first define black-white pebbling game and essential proofs.

Definition 5.11. *(Black-White Pebbling Game) Let $G = (V, E)$ be a circuit-graph with a unique sink s. The game is played on G with two types of pebbles called black pebbles and white pebbles. The game starts with no pebbles on G, following the rules of pebbling game, the goal is to pebble the unique sink s either by a black or a white pebble and finish with no pebbles on G.*

To be precise, at any point in the game, some vertices of G will have pebbles on them (one pebble per vertex), where some of the pebbles are black and some are white. A configuration $\mathcal{C}$ is a subset of vertices, consisting just those vertices that have pebbles on them, each labeled by B, W according to the color of the pebble. That is, $\mathcal{C} = \{(\mathcal{B}, \mathcal{W}) : \mathcal{B}, \mathcal{W} \subseteq V$ with black, white pebbles respectively and $\mathcal{B} \cap \mathcal{W} = \phi\}$. The rules of the pebble game are as follows:

1. *A white pebble may be placed on any vertex.*
2. *A black pebble may be removed from any vertex.*
3. *If all immediate predecessors of a white-pebbled vertex v have pebbles (either B or W) on them, the white pebble can be removed from v.*
4. *If all immediate predecessors of a vertex have a pebble (either B or W) on them, a black pebble may be placed on that vertex.*

A legal black-white pebbling of G is a sequence of configurations $\{\mathcal{C}_0, \ldots, \mathcal{C}_l\}$, where $\mathcal{C}_0 = \mathcal{C}_l = \phi$, there exists a k, $1 \leq k \leq l$ such that $s \in \mathcal{C}_k$ (i.e, $s \in \mathcal{B}_k$ or $s \in \mathcal{W}_k$), and $\mathcal{C}_t$ follows from $\mathcal{C}_{t-1}$ by one of the above rules. The number of pebbles used in such a legal pebbling is the maximal number of pebbles used by any configuration $\mathcal{C}_t$. The black-white pebbling number of G, denoted $BW\text{-}Peb(G)$, is the minimum number of pebbles needed in any legal black-white pebbling of G.

Definition 5.12. *(Essential proofs) Let π be a resolution proof of an unsatisfiable CNF formula F and $\sigma = F_0, \ldots, F_l$ a (clause\variable) space proof conforming to π (see chapter 4 for definitions), we define essential clauses in F_i's by backward induction.*

1. *Let l be the first step such that $\square \in F_l$. Then $\square$ is essential at step l.*

2. *If C is an essential clause at step i, and was inferred in the i^{th} step from the assumptions D, E, then D, E are essential at step $i-1$.*

3. *If $C \in F_i$ is essential at step i then C is essential at step $i-1$, provided $C \in F_{i-1}$. That is, if $C \in F_i$ is neither an axiom download nor derived at step i, and is essential at step i then C must belongs to F_{i-1} and is essential at step $i-1$.*

A (clause\variable) space proof σ is called essential if all clauses in it are essential at all steps.

Observation 5.13. *Removing non-essential clauses from a (clause\variable) space proof conforming with π gives an essential (clause\variable) space proof that conforms with π without increasing the (clause\variable) space, width or size.*

Using Observation 5.13 we can assume from now that all our space oriented proofs are essentials. For F a CNF formula, (i.e, a set of clauses) define $Vars(F)$ to be the set of variables appearing in clauses of F.

Observation 5.14. *For $\sigma = F_0, \ldots, F_l$ an essential space proof of F. The following statements holds:*

1. *l is the first step such that $\square$ appears in F_l, and $F_l = \{\square\}$.*

   Explanation *From Definition 5.12, $\square \in F_l$ is an essential clause. F_l contains only the empty clause because none of the clauses from F_l can derive this empty clause, hence they can not be essential.*

2. *As there are no non-essential clauses in any F_i's, F_{i+1} is obtained from F_i only by the following two rules:*

   Axiom download $F_{i+1} = F_i \cup C$, *where* $C \in F$.

   Inference $F_{i+1} = F_i \cup C \backslash \mathcal{D}$, *where* $A, B \in F_i$, $\frac{A \quad B}{C}$ *and* $\mathcal{D} \subseteq \{A, B\}$

   Explanation *Suppose clauses $C_1, C_2 \in F_i$ and $C_1, C_2 \notin F_{i+1}$. Since each F_i's consists of only essential clauses, both C_1, C_2 are essential in F_i but not in F_{i+1}. Since essential clauses are defined by backward induction, it must be the case that there exists a clause $C_3 \in F_{i+1}$ such that, $\frac{C_1 \quad C_2}{C_3}$. Thus in essential space proof erasure happens only during inference rule. So we conclude that there are no arbitrary erase rule.*

3. *If x is the variable resolved in the inference step $F_i \rightsquigarrow F_{i+1}$, then*

$$Vars(F_i) - \{x\} \subseteq Vars(F_{i+1}) \subseteq Vars(F_i)$$

   Moreover, occurrences of literals corresponding to variables other than x are intact by the step. That is, for variable $y \neq x$, if y appers in F_i then y appears in F_{i+1} as well, and if $\neg y$ appears in F_i then $\neg y$ appears in F_{i+1} as well.

   Explanation *Let $\frac{A \quad B}{C}$ be the resolution rule applied in $(i+1)^{th}$ step such that $A, B \in F_i$ and $C \in F_{i+1}$. Clearly $C = A \cup B - \{x, \neg x\}$. Hence even after erasing both the clauses A, B from F_i to get F_{i+1} and assuming that variable*

x belongs to none of the clauses in F_i other than clauses A and B, the only differences in the variable set (i.e, $Vars(F_i)$ and $Vars(F_{i+1})$) is in the variable x.

Moreover, clauses other than A, B of F_i remains intact in F_{i+1} and hence their literals remains intact as well. Also, literals other than $\neg x, x$ which belongs to the clauses A, B remains intact in F_{i+1} via the clause C.

Now we are ready to prove part 4 of Theorem 5.3 and claim that the following Theorem is sufficient to prove it.

Theorem 5.15. *[7] For any circuit graph G we have $Vspace(\vdash Peb_G) \geq BW\text{-}Peb(G)$.*

Before proving Theorem 5.15, we first show how it implies part 4 of Theorem 5.3.

$$\begin{aligned}
\Omega\Big(\frac{n}{\log n}\Big) &= BW\text{-}Peb(\tilde{G}_n), \text{ by Theorem 5.10} \\
&\leq Vspace(\vdash Peb_{\tilde{G}_n}), \text{ by Theorem 5.15} \\
&\leq Vspace(\pi), \quad \forall \pi \vdash_{tl} Peb_{\tilde{G}_n}, \text{ by definition} \\
&\leq w(\pi) Cspace(\pi), \quad \text{from Lemma 4.5}
\end{aligned}$$

Thus we have the desired result: for any tree-like resolution proof π of $Peb_{\tilde{G}_n}$, we have, $w(\pi)Cspace(\pi) \geq \Omega(\frac{n}{\log n})$.

And from Remark 4.12, for any tree-like resolution $\pi \vdash_{tl} F$, we have $Cspace(\pi) \leq \log_2 |\pi| + 1$, Thus for any $\pi \vdash_{tl} Peb_{\tilde{G}_n}$, we have, $w(\pi)Cspace(\pi) \leq w(\pi) \log_2 |\pi|$, which proofs the first inequality of part 4 as well.

Proof of Theorem 5.15

We will show that any essential space proof $\sigma = F_0, \ldots, F_l$ of Peb_G is actually a legal black-white pebbling $\mu = \{\mathcal{C}_0, \ldots, \mathcal{C}_l\}$ of G with $|\mathcal{C}_i| = |\mathcal{B}_i| + |\mathcal{W}_i| \leq lit(F_i)$, i.e, number of pebbles used in i^{th} configuration is $\leq lit(F_i)$. This will proof the required claim. Let $\sigma = F_0, \ldots, F_l$ be an essential space proof of Peb_G. For F a set of clauses define $Positive(F) \subseteq Vars(F)$ to be the set of variables that have at least one appearance as a positive literals in F. Now define the following pebbling sequence in G

$$\mathcal{C}_t(v) = \begin{cases} \text{Black} & x_v \in Positive(F_t) \\ \text{White} & x_v \in Vars(F_t) - Positive(F_t) \\ \text{Null} & \text{Otherwise} \end{cases}$$

Clearly by its definition, number of pebbles in $\mathcal{C}_i \leq lit(F_i)$ for each i. Therefore μ is using no more than $Vspace(\sigma)$ pebbles. Now we need to show that μ is a legal black-white pebbling of G. As $F_0 = \phi$ and $F_l = \{\square\}$ we have, $\mathcal{C}_0 = \phi = \mathcal{C}_l$. Since $Peb_G - (\neg x_s)$ is satisfiable, any resolution proof of Peb_G must use the sink axiom $\neg x_s$, hence at some point in μ a pebble will be placed on s. Thus the only thing remains to show is that $\mathcal{C}_t$ follows from $\mathcal{C}_{t-1}$ using a valid black-white pebbling rule. We will show this by induction on $t = 0, \ldots, l$ and split the proof into cases:

Inference Assume variable x_v was resolved in the t^{th} step (i.e, $F_{t-1} \rightsquigarrow F_t$). By Observation 5.14, it is enough to focus only on x_v. Clearly $x_v \in \mathcal{B}_{t-1}$ (i.e, $\mathcal{C}_{t-1}(v) = \text{Black}$). We have three cases:

Case (1) $x_v \in Positive(F_t)$. Then we are leaving black pebble on v as it is.

Case (2) $x_v \in Vars(F_t) - Positive(F_t)$. In this case we are removing black pebble from v and placing white pebble. A valid step.

Case (3) $x_v \notin Vars(F_t)$. We are removing black pebble. A valid step.

Thus all the steps are legal black-white pebbling steps, showing that the transition $\mathcal{C}_{t-1}$ to $\mathcal{C}_t$ is legal.

Axiom Downloads Recall that there are three types of axioms in the formula Peb_G. Let A be the axiom downloaded in the t^{th} step. There are three cases:

Case (1) A is some source axiom. Let $A = x_v$, for v a source vertex. As $A \in F_t$, we have $x_v \in Positive(F_t)$. There are three cases: first case, $x_v \in Positive(F_{t-1})$, which implies, $v \in \mathcal{B}_{t-1}$. In this case we are leaving black pebble in v as it is. Second case, $x_v \in Vars(F_{t-1}) - Positive(F_{t-1})$, which implies, $v \in \mathcal{W}_{t-1}$. In this case we are removing white pebble and placing black pebble in v. This is a valid step as v is a source vertex. Last case, $x_v \notin Vars(F_{t-1})$, which implies, v has no pebble in $(t-1)^{th}$ step, and we are placing a black pebble, a valid step as v is a source vertex.

Case (2) $A = (\neg x_s)$, the sink axiom. There are three cases: first case, $x_s \in Positive(F_{t-1})$, which implies, $s \in \mathcal{B}_{t-1}$. In this case, $x_s \in Positive(F_t)$ as well, and we are leaving black pebble as it is. A valid step. Second case, $x_s \in Vars(F_{t-1}) - Positive(F_{t-1})$, which implies, $s \in \mathcal{W}_{t-1}$. In this case, $x_s \in Vars(F_t) - Positive(F_t)$, and we are leaving white pebble as it is. A valid step. Last case, $x_s \notin Vars(F_{t-1})$, which implies, s has no pebble in $(t-1)^{th}$ step. In this case, as $A \in F_t$, $x_s \in Vars(F_t) - Positive(F_t)$, and we are placing a white pebble on s. A valid step.

Case (3) A is some pebbling axiom. Let $A = (\neg x_u \vee \neg x_w \vee x_v)$. Look at the vertex u. There are three cases: first case, $x_u \in Positive(F_{t-1})$, which implies, $u \in \mathcal{B}_{t-1}$. In this case, $x_u \in Positive(F_t)$ as well. Leave the black pebble as it is. Second case, $x_u \in Vars(F_{t-1}) - Positive(F_{t-1})$, which implies, $u \in \mathcal{W}_{t-1}$. In this case, $x_u \in Vars(F_t) - Positive(F_t)$ as well. And we are leaving white pebble as it is. A valid step. Last case, $x_u \notin Vars(F_{t-1})$, which means, u has no pebble in $(t-1)^{th}$ step. Now as $A \in F_t$, we have $x_u \in Vars(F_t) - Positive(F_t)$. So we are placing a white pebble on u. A valid step. Similarly, we repeat for vertex w. After this steps, we are sure that both the vertex u and w has some pebble on it. Now look at vertex v. As $A \in F_t$, we have $x_v \in Positive(F_t)$. There are three cases: first case, $x_v \in Positive(F_{t-1})$, which implies $v \in \mathcal{B}_{t-1}$. In this case, we are leaving black pebble on v as it is. Second case, $x_v \in Vars(F_{t-1}) - Positive(F_{t-1})$, which implies, $v \in \mathcal{W}_{t-1}$. In this case, we are removing white pebble from v and placing black pebble on v. This is a valid step, since both of its children u, w, are already pebbled at this moment. Last case, $x_v \notin Vars(F_{t-1})$. In this case, we are placing a black pebble on v. A valid step, as both of its children are already pebbled.

So again all the steps are legal black-white pebbling steps, showing that the transition $\mathcal{C}_{t-1}$ to $\mathcal{C}_t$ is legal. This completes the proof of Theorem 5.15.

As pointed out at the beginning of the Chapter, the paper [7] also contains tradeoff results for general resolution. In particluar, Ben-Sasson [7] showed that there exists contradictions that can not have small size and small clause space simultaneously:

Theorem 5.16. *[7] For infinitely many integers n, there exists unsatisfiable CNF formulas F_n of size (number of clauses) n such that:*

1. *$w(\vdash F_n) = O(1)$ and $S(\vdash F_n) = O(n)$. That is, there exists a resolution proof of F_n with linear size and constant width simultaneously.*
2. *$Vspace(\vdash F_n) = \Omega(\frac{n}{\log n})$.*
3. *For any general resolution proof π of F_n we have,*

$$Cspace(\pi) \log |\pi| = \Omega(\frac{n}{\log n}).$$

Chapter 6

Conclusion

In this report we have seen resolution and tree-like resolution proof system. We have defined three complexity measures for resolution: size, width and (clause\variable) space. We have seen lower-bound results for resolution. In particular we saw in Theorem 3.1 that any resolution proof for pigeonhole principle with n pigeons and $n-1$ holes, must have size at least $2^{n/20}$. We also saw width-size relationship in Corollary 3.12. The main point of this result is, now to prove size lower bound it is enough to prove width lower bound. We saw in Corollary 3.19 that there are formulas with constant width, having polynomial size resolution proof but requires exponential size in tree-like resolution proof system.

For tree-like resolution proofs we have seen relationship between size and clause space. In particular, Theorem 4.11 shows that lower bound in clause space implies lower bound in size for tree-like resolution system. We have seen combinatorial characterization of resolution width defined by Atserias and Dalmau [5]. We have also seen game characterization of tree-like clause space defined by Impagliazzo and Pudl*á*k [20].

At the end we have seen tradeoffs results for tree-like resolutions, proved by Ben-Sasson [7]. In particular, we have seen in Theorem 5.3 that there exists unsatisfiable CNF formulas (pebbling contradictions), such that it has constant width resolution proof, tree-like resolution proof with linear size and constant space, but any tree-like resolution proof π of pebbling contradiction must have $w(\pi)Cspace(\pi) = \Omega(\frac{n}{\log n})$. That is, optimizing both width and clause space simultaneously for pebbling contradictions is not possible in tree-like resolution system.

Appendix A

Complexity Classes and Useful Notations Used in the Report

Complexity Classes

We say that a machine decides a language $L \in \{0,1\}^*$ if it computes the function $f_L : \{0,1\}^* \to \{0,1\}$, where $f_L(x) = 1 \Leftrightarrow x \in L$.

The class $DTime$
Let $T : \mathbf{N} \to \mathbf{N}$ be some function. A language L is in $DTime(T(n))$ iff there is a "deterministic" Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L.

The class P
$P = \cup_{c \geq 1} DTime(n^c)$.

The class NP
A language $L \in \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbf{N} \to \mathbf{N}$ and a polynomial time TM $\mathbf{M}$ (called verifier for L) such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{s.t.} M(x,u) = 1$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a certificate for x (with respect to the language L and machine M).

The class $coNP$
If $L \in \{0,1\}^*$ is a language, then we denote by $\bar{L}$ the complement of L. That is $\bar{L} = \{0,1\}^* \backslash L$.

$$coNP = \{L : \bar{L} \in NP\}$$

$coNP$, Alternative Definition
For every $L \in \{0,1\}^*$, we say that $L \in coNP$ if there exists a polynomial $p : \mathbf{N} \to \mathbf{N}$ and a polynomial time TM $\mathbf{M}$ such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \forall u \in \{0,1\}^{p(|x|)}, M(x,u) = 1$$

Notations

Let F be an unsatisfiable CNF formula, π a resolution proof of F and C be some clause. Then we have the following notations:

Notations related to C and F

- $width(C)$: number of literals in C.
- $|F|$: number of clauses in F.
- $w(F) = \max_{C \in F}\{width(C)\}$: width of a largest clause in F.
- $lit(F) = \sum_{C \in F}\{width(C)\}$: sum of width of all clauses in F.

Notations related to π

- $\pi \vdash F$: π is a general resolution proof of F.
- $\pi \vdash_{tl} F$: π is a tree-like resolution proof of F.
- $|\pi|$: number of clauses in π.
- $S(\vdash F) = \min_{\pi \vdash F} |\pi|$: size of deriving F in general resolution proof.
- $S_T(\vdash F) = \min_{\pi \vdash_{tl} F} |\pi|$: size of deriving F in tree-like resolution proof.
- $w(\pi) = \max_{C \in \pi}\{width(C)\}$: width of a largest clause in π.
- $w(\vdash F) = \min_{\pi \vdash F}\{w(\pi)\}$: width of deriving F in general resolution.
- $w(\vdash_{tl} F) = \min_{\pi \vdash_{tl} F}\{w(\pi)\}$: width of deriving F in tree-like resolution.
- G_π: proof graph (directed acyclic graph) corresponding to π.
- $Cspace(\pi)$: (clause space of π) pebbling number of G_π (for pebbling game see Definition 4.2).
- $Cspace(\vdash F) = \min_{\pi \vdash F}\{\text{pebbling number of } G_\pi\}$: clause space of deriving F in general resolution.
- $Cspace(\vdash_{tl} F) = \min_{\pi \vdash_{tl} F}\{\text{pebbling number of } G_\pi\}$: clause space of deriving F in tree-like resolution.
- $Vspace(\pi)$: (variable space of π) Vpebbling number of G_π (for Vpebbling game see discussions after Lemma 4.3).
- $Vspace(\vdash F) = \min_{\pi \vdash F}\{\text{Vpebbling number of } G_\pi\}$: variable space of deriving F in general resolution.
- $Vspace(\vdash_{tl} F) = \min_{\pi \vdash_{tl} F}\{\text{Vpebbling number of } G_\pi\}$: variable space of deriving F in tree-like resolution.

Bibliography

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

[2] Cook S. A. and Reckhow A. R. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1977.

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

[4] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002.

[5] A. Atserias and V. Dalmau. A combinatorial characterization of resolution width. *18th IEEE Conference on Computational Complexity*, pages 239–247, 2003.

[6] Paul Beam. Proof complexity. *IAS/Park City Mathematics Institute, Volume 10, Computational Complexity Theory*, 2000.

[7] Eli Ben-Sasson. Size space tradeoffs for resolution. *In Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 457–464, May 2002.

[8] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of treelike and general resolution. *Combinatorica*, 24(4):585–603, September 2004.

[9] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow − resolution made simple. *In Proceedings of the Thirty first Annual ACM Symposium on Theory of Computing, Atlanta, GA*, pages 517–526, May 1999.

[10] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like resolution size. *Electronic Colloquium on Computational Complexity (ECCC)*, 19, 2012.

[11] Maria Luisa Bonet and Nicola Galesi. A study of proof search algorithms for resolution and polynomial calculus. *In Proceedings 40th Annual Symposium on Foundations of Computer Science, New York, NY, IEEE*, October 1999.

[12] J. L. Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001.

[13] J. L. Esteban and Jacobo Torán. Combinatorial characterization of tree-like space. *Information Processing Letters*, 87(6):295–300, 2003.

[14] Zvi. Galil. On resolution with clauses of bounded size. *SIAM Journal on Computing*, 6(3):444–459, 1977.

[15] John R. Gilbert and Robert Endre Tarjan. Variations of a pebble game on graphs. *Technical Report STAN-CS-78-661, Stanford University*, 1978.

[16] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[17] P. G. Kolaitis and M. Y. Vardi. On the expressive power of datalog: tools and a case study. *Journal of Computer and System Sciences*, 51:110–134, 1995.

[18] Jan Krajiček. Propositional proof complexity I. *Mathematical Institute, Academic of Sciences of the Czech Republic Praha*, 2003.

[19] Davis M. and Putnam H. A computing procedure for quatification theory. *Journal of the ACM*, 7(3):210–215, 1960.

[20] P. Pudlák and R. Impagliazzo. A lower bound for DLL algorithms for k-SAT. *Proc.11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 128–136, 2000.

[21] U. Schoning. Logic for computer scientist. *Birkhauser*, 1989.

[22] Jacobo Torán. Space and width in propositional resolution. *Bulletin of the European Association for Theoretical Computer Science (EATCS) 83*, 2004.

[23] Jacobo Torán. Lower bounds for space in resolution. *In Proceedings of the 13th International Workshop on Computer Science Logic (CSL '99), Volume 1683 of Lecture Notes in Computer Science*, pages 362–373, Springer 1999.