Chapter 4

Proof Complexity and Resolution

In this chapter, we give a very brief overview of some of the central concepts in proof complexity. We then proceed to define the resolution proof system and state the results mentioned in Chapter 2, as well as some other results relevant to this thesis, in a more formal setting. As already noted, we refer to, for instance, the books [12, 28, 30] or the survey papers [15, 76, 83] for more details.

4.1 A Proof Complexity Primer

We assume the existence of an infinite set *Vars* of boolean (or propositional logic) variables ranging over $\{0, 1\}$, where we identify 0 with *FALSE* and 1 with *TRUE*, respectively. We use the traditional set of logical connectives: negation $\neg$, conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$ and bi-implication (or equivalence) $\leftrightarrow$.

The set PROP of propositional logic formulas is the smallest set X such that

- $x \in X$ for all propositional logic variables $x \in \mathit{Vars}$,
- if $F, G \in X$ then $\big(F \wedge G\big), \big(F \vee G\big), \big(F \rightarrow G\big), \big(F \leftrightarrow G\big) \in X$,
- if $F \in X$ then $\big(\neg F\big) \in X$.

We write $\mathit{Vars}(F)$ to denote the set of variables of a formula F, i.e., $\mathit{Vars}(x) = \{x\}$, $\mathit{Vars}(\neg F) = \mathit{Vars}(F)$, $\mathit{Vars}(F \wedge G) = \mathit{Vars}(F) \cup \mathit{Vars}(G)$, and analogously for the other connectives.

Let α denote a truth value assignment, i.e., a function $\alpha : \mathit{Vars} \mapsto \{0, 1\}$. Then α is extended from variables to formulas in the canonical way by defining that $\alpha(\neg F) = 1$ if $\alpha(F) = 0$, $\alpha(F \wedge G) = 1$ if $\alpha(F) = \alpha(G) = 1$, $\alpha(F \vee G) = 1$ unless $\alpha(F) = \alpha(G) = 0$, $\alpha(F \rightarrow G) = 1$ unless $\alpha(F) = 1$ and $\alpha(G) = 0$, and $\alpha(F \leftrightarrow G) = 1$ if $\alpha(F) = \alpha(G)$. We say that F is

- *satisfiable* if there is an assignment α with $\alpha(F) = 1$,
- *valid* or *tautological* if all assignments satisfy F,

- *falsifiable* if there is an assignment α with $\alpha(F) = 0$,
- *unsatisfiable* or *contradictory* if all assignments falsify F.

If an assignment α satisfies a formula F, α is called a *model* of F. If α falsifies F, α is called a *counter-model*. The set of all tautological propositional logic formulas (or *tautologies*) F is denoted TAUTOLOGY. For more details, see [35] or any other standard textbook on logic.

The definition below from [12] is an adaption of the original definition in [33].

Definition 4.1 (Proof system). A *proof system* for a language L (or set L, depending on which terminology one prefers) is a polynomial-time algorithm P such that

1. for all $x \in L$ there is a string π (a *proof*) such that $P(x, \pi) = 1$,
2. for all $x \notin L$ and for all strings π it holds that $P(x, \pi) = 0$.

Note that P does not have to be polynomial-time in x only. If the proof π is large, P can use time polynomial in the size of the proof while checking it.

Let us define the *size* $S(x)$ of a string x to be the number of symbols in x. Then the *complexity* of a proof system P for a language L, which we denote $cplx(P)$, is the smallest bounding function $g : \mathbb{N} \mapsto \mathbb{N}$ such that every $x \in L$ has a proof of size at most $g\bigl(S(x)\bigr)$, or in more formal notation

$$x \in L \Leftrightarrow \exists \pi \; S(\pi) \leq g\bigl(S(x)\bigr) \;\wedge\; P\bigl(x, \pi\bigr) = 1 \;\; . \tag{4.1}$$

If a proof system is of polynomial complexity, it is said to be *polynomially bounded* or *p-bounded*. Thus, NP is exactly the set of languages with polynomially bounded proof systems.

In this thesis, we are interested in proof systems for the set of all tautologies in propositional logic.

Definition 4.2 (Propositional proof system). A *propositional proof system* P is a proof system for TAUTOLOGY.

That is, a propositional proof system is a polynomial-time computable binary predicate P satisfying the following property: for all propositional logic formulas F it holds that $F \in$ TAUTOLOGY if and only if there exists a proof π of F such that $P\bigl(F, \pi\bigr)$ is true.

A quite common variation of this theme, a variation that we will focus on in the rest of this thesis, is to prove instead that formulas in conjunctive normal form (CNF formulas) are unsatisfiable. The reason that this is essentially the same problem is that it is possible to convert any propositional logic formula F to a CNF formula in such a way that it has only linearly larger size and is unsatisfiable if and only if the original formula is a tautology. One example of such a conversion is a transformation first used by Tseitin [81]. The idea in Tseitin's transformation is

$$G \doteq H_1 \wedge H_2 : \qquad Tr\big(G\big) = \begin{aligned}[t] &\big(\neg x_G \vee x_{H_1}\big) \\ \wedge\ &\big(\neg x_G \vee x_{H_2}\big) \\ \wedge\ &\big(x_G \vee \neg x_{H_1} \vee \neg x_{H_2}\big) \end{aligned}$$

$$G \doteq H_1 \vee H_2 : \qquad Tr\big(G\big) = \begin{aligned}[t] &\big(\neg x_G \vee x_{H_1} \vee x_{H_2}\big) \\ \wedge\ &\big(x_G \vee \neg x_{H_1}\big) \\ \wedge\ &\big(x_G \vee \neg x_{H_2}\big) \end{aligned}$$

$$G \doteq H_1 \rightarrow H_2 : \qquad Tr\big(G\big) = \begin{aligned}[t] &\big(\neg x_G \vee \neg x_{H_1} \vee x_{H_2}\big) \\ \wedge\ &\big(x_G \vee x_{H_1}\big) \\ \wedge\ &\big(x_G \vee \neg x_{H_2}\big) \end{aligned}$$

$$G \doteq H_1 \leftrightarrow H_2 : \qquad Tr\big(G\big) = \begin{aligned}[t] &\big(\neg x_G \vee \neg x_{H_1} \vee x_{H_2}\big) \\ \wedge\ &\big(\neg x_G \vee x_{H_1} \vee \neg x_{H_2}\big) \\ \wedge\ &\big(x_G \vee \neg x_{H_1} \vee \neg x_{H_2}\big) \\ \wedge\ &\big(x_G \vee x_{H_1} \vee x_{H_2}\big) \end{aligned}$$

Figure 4.1: Tseitin's transformation to CNF formulas.

to introduce a new variable x_G for each subformula $G \doteq H_1 \circ H_2$ in F, where we let $\circ$ denote one of the connectives $\wedge$, $\vee$, $\rightarrow$, or $\leftrightarrow$, and use $\doteq$ to denote syntactic equality. The formula F is then translated to conjunctive normal form by adding a set of clauses $Tr\big(G\big)$ for each subformula G which enforces that the the truth value of x_G is computed correctly given the truth values of x_{H_1} and x_{H_2}. These clauses $Tr\big(G\big)$ are presented in Figure 4.1. Finally, a unit clause $\neg x_F$ is added. It is easy to verify that the resulting CNF formula is unsatisfiable if and only if F is a tautology. In this way, any sound and complete system which produces refutations of formulas in conjunctive normal form can be considered as a general propositional proof system.

We have already argued that proving tautologies (or equivalently, as we have just seen, refuting unsatisfiable CNF formulas) is an important applied problem, but one other reason why proof complexity is interesting from a theoretical point of view is the following theorem.

Theorem 4.3 ([33]). *NP = co-NP if and only if there exists a polynomially bounded propositional proof system.*

Proof. ($\Rightarrow$) Obviously, TAUTOLOGY $\in$ $\mathsf{co\text{-}NP}$ since F is *not* a tautology if and only if $\neg F \in$ SATISFIABILITY. If $\mathsf{NP} = \mathsf{co\text{-}NP}$, then TAUTOLOGY $\in \mathsf{NP}$ has a polynomially bounded proof system by definition.

($\Leftarrow$) Conversely, assume that there exists a p-bounded propositional proof system. Then TAUTOLOGY $\in \mathsf{NP}$, and since TAUTOLOGY is complete for $\mathsf{co\text{-}NP}$ it follows that $\mathsf{NP} = \mathsf{co\text{-}NP}$. □

Since P is closed under complement, we have the following immediate corollary.

Corollary 4.4. *If all propositional proof systems have superpolynomial complexity, then $\mathsf{P} \neq \mathsf{NP}$.*

The conventional wisdom is that it should hold that $\mathsf{NP} \neq \mathsf{co\text{-}NP}$, but Corollary 4.4 explains why a proof of this still appears to be light years away. One line of research in proof complexity is to try to approach this distant goal by studying successively stronger propositional proof systems and relating their strengths. In this context, *polynomial simulations*, or *p-simulations*, play an important role.

Definition 4.5 (p-simulation). A propositional proof system P_1 *polynomially simulates*, or *p-simulates*, another propositional proof system P_2 if there exists a polynomial-time computable function f such that for all $F \in$ TAUTOLOGY it holds that $P_2\big(F, \pi\big) = 1$ if and only if $P_1\big(F, f\big(\pi\big)\big) = 1$.

If the complexity of two proof systems are within polynomial factors, we consider them to be "equally strong" for theoretical purposes.

Definition 4.6 (p-equivalence). Two propositional proof systems P_1 and P_2 are *polynomially equivalent*, or *p-equivalent*, if each proof system p-simulates the other.

Polynomial simulations define a partial order relation on proof systems. A natural question is whether there is a maximal element with respect to this ordering or not. This is not known, and there is little circumstantial evidence either way. Formally, let us say that a propositional proof system is *p-optimal* if it p-simulates every other propositional proof system. Then we have the following result.

Theorem 4.7 ([52]). *If $\mathsf{EXP} = \mathsf{NEXP}$, there is a p-optimal propositional proof system.*

This does not tell us too much, though, since this complexity class equality is considered implausible.

The definitions so far say nothing about how hard it might be to actually *find* proofs in the proof system P. Let us say that a *proof search algorithm A_P* for P is a deterministic algorithm A_P that takes as input a formula F and generates a proof π of F in the format specified by the proof system P (i.e., such that $P\big(F, \pi\big) = 1$) if F is valid and reports that F is falsifiable otherwise. Then the following definition from [12] captures a property that we would like our propositional proof system to have.

Definition 4.8 (Automatizability). Given a propositional proof system P and a function $f : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$, we say that P is $f(n, S)$-*automatizable* if there exists a proof search algorithm A_P such that if $F \in$ TAUTOLOGY, then A_P on input F outputs a P-proof of F in time at most $f(n, S)$, where n is the size of F and S is the size of a smallest P-proof of F.

A proof system P is called *automatizable* if it is $f(n, S)$-automatizable for some $f(n, S) = \text{poly}(n) \cdot \text{poly}(S)$. The proof system P is *quasi-automatizable* if it is $f(n, S)$-automatizable for $f(n, S) = n^{c_1} \cdot \exp\bigl(\log^{c_2} S\bigr)$ for some constants c_1, c_2.

Note that automatizability seems to be the right definition because given a proof system P, this is in a sense the best we can hope for. If there are no small proofs of F in P to be found, then no proof search algorithm A_P in P can be expected to find proofs quickly. However, given a bound on the best any proof search algorithm for P can do, we want an algorithm A_P that performs well with respect to this bound.

Let us conclude this very brief introduction to proof complexity by giving examples of concrete propositional proof systems. No introduction to proof complexity can be complete without at least mentioning what a Frege system is. The next two definitions are (slightly adapted) from [28].

Definition 4.9 (Frege system). Let $F, F_1, \ldots, F_k$ be propositional logic formulas over the variables $x_1, \ldots, x_n$. A *Frege rule* is a pair

$$\bigl(\bigl\{F_1(x_1, \ldots, x_n), \ldots, F_k(x_1, \ldots, x_n)\bigr\}, F(x_1, \ldots, x_n)\bigr)$$

such that the implication $F_1(x_1, \ldots, x_n) \wedge \ldots \wedge F_k(x_1, \ldots, x_n) \rightarrow F(x_1, \ldots, x_n)$ is a tautology. Usually the rule is written as $\frac{F_1, \ldots, F_k}{F}$. A Frege rule with zero assumptions is called an *axiom schema*.

A Frege rule is applied by substituting arbitrary formulas for the variables $x_1, \ldots, x_n$. A *Frege proof* of a formula G is a sequence of formulas such that each formula follows from previous ones by an application of a Frege rule from a given set of rules and the last formula is G.

A *Frege system* $\mathfrak{F}$ is determined by a finite complete set of connectives B and a finite set of Frege rules such that $\mathfrak{F}$ is implicationally complete for the set of formulas over the basis B.

If Definition 4.9 seems very relaxed, it is because the details do not matter very much.

Theorem 4.10 ([33]). *Any two Frege systems p-simulate each other.*

Sadly, there are currently no strong lower bounds known for Frege systems (but see [27] for a survey of what is known). However, by restricting the model, somewhat similarly to what is done in circuit complexity, we get subsystems for which it is known how to prove superpolynomial lower bounds.

Definition 4.11 (Bounded-depth Frege system). Consider formulas in basis $\{\wedge, \vee, \neg\}$, The depth of a formula is the maximum number of alterations of connectives in it. A *depth-d Frege proof* is a Frege proof where all formulas in the proof sequence have depth at most d.

Theorem 4.12 ([51, 69]). *The pigeonhole principle formulas (encoding the statement that if $n+1$ pigeons are placed in n pigeonholes, then at least one pigeonhole must contain more than one pigeon) require bounded-depth Frege proofs of size growing exponentially in n.*

Informally speaking, there seems to be an unfortunate trade-off for proof systems in that if a proof system is sufficiently powerful, then it is not automatizable. For instance, bounded-depth Frege systems are not automatizable under plausible cryptographic assumptions. More formally, we call $n \in \mathbb{N}$ a *Blum integer* if $n = pq$ for primes $p \equiv q \equiv 3 \pmod 4$. Then the following theorem is known.

Corollary 4.13 ([23]). *If factoring Blum integers is hard, then any proof system that can p-simulate bounded-depth Frege is not automatizable.*

The resolution proof systems, that we define next, can be viewed as a very limited form of a bounded-depth Frege system, namely depth-0 Frege. Even this proof system is likely not to be automatizable [6], but as was mentioned in Chapter 2 there are proof search algorithms for resolution that seem to work very well in practice.

4.2 Definition of the Resolution Proof System

A *literal* is either a propositional logic variable x or its negation, which we will from now on denote $\overline{x}$. Sometimes, though, it will be convenient to write x^1 for x and x^0 for $\overline{x}$. We define $\overline{\overline{x}} = x$. Two literals a and b are *strictly distinct* if $a \neq b$ and $a \neq \overline{b}$, i.e., if they refer to distinct variables.

A *clause* $C = a_1 \vee \cdots \vee a_k$ is a set of literals. Throughout this thesis, all clauses C are assumed to be nontrivial in the sense that all literals in C are pairwise strictly distinct (otherwise C is trivially true). We say that C is a *subclause* of D if $C \subseteq D$. A clause containing at most k literals is called a *k-clause*.

A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a set of clauses. A *k-CNF formula* is a CNF formula consisting of k-clauses. We define the *size* $S(F)$ of the formula F to be the total number of literals in F counted with repetitions. More often, we will be interested in the number of clauses $|F|$ of F.

In this thesis, when nothing else is stated it is assumed that A, B, C, D denote clauses, $\mathbb{C}, \mathbb{D}$ sets of clauses, x, y propositional variables, a, b, c literals, α, β truth value assignments and ν a truth value 0 or 1. We write

$$\alpha^{x=\nu}(y) = \begin{cases} \alpha(y) & \text{if } y \neq x, \\ \nu & \text{if } y = x, \end{cases} \tag{4.2}$$

$$
\begin{aligned}
F = \quad & (x \lor z) \land (\overline{z} \lor y) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u}) \\
& \land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})
\end{aligned}
$$

(a) CNF formula F.

1.	$x \lor z$	Axiom	9.	$x \lor y$	Res$(1,2)$
2.	$\overline{z} \lor y$	Axiom	10.	$x \lor \overline{y}$	Res$(3,4)$
3.	$x \lor \overline{y} \lor u$	Axiom	11.	$\overline{x} \lor u$	Res$(5,6)$
4.	$\overline{y} \lor \overline{u}$	Axiom	12.	$\overline{x} \lor \overline{u}$	Res$(7,8)$
5.	$u \lor v$	Axiom	13.	x	Res$(9,10)$
6.	$\overline{x} \lor \overline{v}$	Axiom	14.	$\overline{x}$	Res$(11,12)$
7.	$\overline{u} \lor w$	Axiom	15.	0	Res$(13,14)$
8.	$\overline{x} \lor \overline{u} \lor \overline{w}$	Axiom			

(b) Resolution refutation of F.

Figure 4.2: Example resolution refutation.

to denote the truth value assignment that agrees with α everywhere except possibly at x, to which it assigns the value ν. We let $\mathit{Vars}(C)$ denote the set of variables and $\mathit{Lit}(C)$ the set of literals in a clause C.[1] This notation is extended to sets of clauses by taking unions. Also, we employ the standard notation $[n] = \{1, 2, \ldots, n\}$.

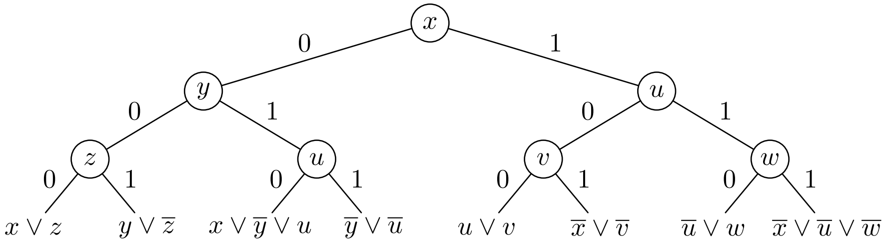
A *resolution derivation* $\pi : F \vdash A$ of a clause A from a CNF formula F is a sequence of clauses $\pi = \{D_1, \ldots, D_\tau\}$ such that $D_\tau = A$ and each line D_i, $i \in [\tau]$, either is one of the clauses in F (*axioms*) or is derived from clauses D_j, D_k in π with $j, k < i$ by the *resolution rule*

$$\frac{B \lor x \quad C \lor \overline{x}}{B \lor C} \ . \tag{4.3}$$

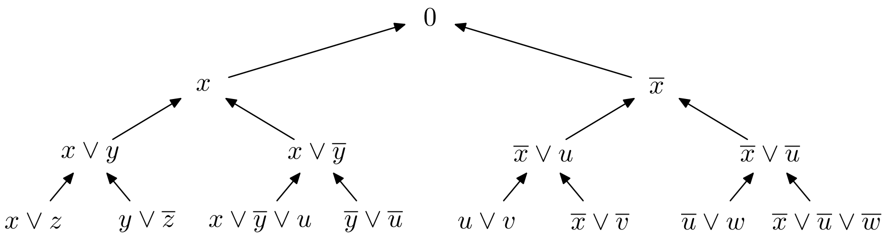
We refer to (4.3) as *resolution on the variable* x and to $B \lor C$ as the *resolvent* of $B \lor x$ and $C \lor \overline{x}$ on x. A *resolution refutation* π of a CNF formula F is a resolution derivation of the empty clause 0, i.e., the clause with no literals, from F. See Figure 4.2 for an example resolution refutation. Perhaps somewhat confusingly, π is sometimes also referred to as a *resolution proof* of F in the literature, since we can view F as being the encoding of the negation of a tautology as explained in Section 4.1. In this thesis, we will try to stick to talking about "refutations of F," but the terms "resolution refutation" and "resolution proof" in general will be used interchangeably.

For a formula F and a set of formulas $\mathcal{G} = \{G_1, \ldots, G_n\}$, we say that $\mathcal{G}$ *implies* F, denoted $\mathcal{G} \vDash F$, if every truth value assignment satisfying all formulas

[1] Although the notation $\mathit{Lit}(C)$ is slightly redundant given the definition of a clause as a set of literals, we include it for clarity.



(a) Decision tree for F with internal vertices labelled by variables queried.



(b) Corresponding resolution refutation of F.

Figure 4.3: Proof by example of implicational completeness of resolution.

$G \in \mathcal{G}$ must satisfy F as well. It is well known that resolution is sound and implicationally complete. That is, if there is a resolution derivation $\pi : F \vdash A$, then $F \vDash A$, and if $F \vDash A$, then there is a resolution derivation $\pi : F \vdash A'$ for some $A' \subseteq A$. In particular, F is unsatisfiable if and only if there is a resolution refutation of F.

We note that the soundness is not hard to argue—it follows from the fact that the resolution rule (4.3) is sound. Completeness is not immediately obvious, but let us sketch a proof. Given any unsatisfiable CNF formula F, we can build a decision tree for F, where we query some variable x in each vertex and branch left or right depending on the value assigned to x. Then the paths from the root downwards in the tree correspond to partial truth value assignments, and as soon as an assignment falsifies a clause, we add a leaf labelled by that clause. It is clear that we can build such a decision tree for any unsatisfiable formula F, and if we then turn this decision tree upside down, we have (essentially) a resolution refutation of F. Figure 4.3 gives a proof by example of this fact, and although we omit the details it is not hard to make this into a formal proof.

With every resolution derivation $\pi : F \vdash A$ we can associate a DAG G_π, with the clauses in π labelling the vertices and with edges from the assumption clauses to the resolvent for each application of the resolution rule (4.3). There might be several different derivations of a clause C in π, but if so we can label each occurrence of C with a time-stamp when it was derived and keep track of which copy of C is used where. A resolution derivation π is *tree-like* if any clause in the derivation is used at most once as a premise in an application of the resolution rule, i.e., if G_π is a tree. (We may make different "time-stamped" vertex copies of the axiom clauses in order to make G_π into a tree). As we can see from Figure 4.3(b), our example refutation in Figure 4.2 is tree-like.

Given this definition of the resolution proof system, we can define the *length* $L(\pi)$ of a resolution derivation π as the number of clauses in it, and the *width* $W(\pi)$ of a derivation is the size of a largest clause in it. For instance, the refutation in Figure 4.2 has length 15 and width 3. However, in order to define space in a natural way and to be able to reason about trade-offs between measures, it is convenient to describe resolution is a slightly different way.

Following the exposition in [39], a resolution proof can be seen as a Turing machine computation, with a special read-only input tape from which the axioms can be downloaded and a working memory where all derivation steps are made. Then the *clause space* of a resolution proof is the maximum number of clauses that need to be kept in memory simultaneously during a verification of the proof. The *variable space* is the maximum total space needed, where also the width of the clauses is taken into account. The formal definitions follow.

Definition 4.14 (Resolution ([4])). A *clause configuration* $\mathbb{C}$ is a set of clauses. A sequence of clause configurations $\{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ is a *resolution derivation* from a CNF formula F if $\mathbb{C}_0 = \emptyset$ and for all $t \in [\tau]$, $\mathbb{C}_t$ is obtained from $\mathbb{C}_{t-1}$ by one[2] of the following rules:

Axiom Download $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$ for some $C \in F$.

Erasure $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$ for some $C \in \mathbb{C}_{t-1}$.

Inference $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{D\}$ for some D inferred by resolution from $C_1, C_2 \in \mathbb{C}_{t-1}$.

A resolution derivation $\pi : F \vdash A$ of a clause A from a formula F is a derivation $\{\mathbb{C}_0, \ldots, \mathbb{C}_\tau\}$ such that $\mathbb{C}_\tau = \{A\}$. A *resolution refutation* of F is a derivation of the empty clause 0 from F.

Definition 4.15 (Length, width, space). The *width* $W(C)$ of a clause C is $|C|$, i.e., the number of literals in it. The width of a clause configuration $\mathbb{C}$ is $W(\mathbb{C}) = \max_{C \in \mathbb{C}}\{W(C)\}$. The *clause space* of a configuration $\mathbb{C}$ is $Sp(\mathbb{C}) = |\mathbb{C}|$, i.e., the number of clauses in $\mathbb{C}$, and the *variable space* is $VarSp(\mathbb{C}) = \sum_{C \in \mathbb{C}} W(C)$.

Let π be a resolution derivation. Then:

- The *length* $L(\pi)$ of π is the number of axiom download and inference steps in π.
- The *width* of π is $W(\pi) = \max_{\mathbb{C} \in \pi}\{W(\mathbb{C})\}$.

[2] In some previous papers, resolution is defined so as to allow every derivation step to *combine* one or zero applications of each of the three derivation rules. Therefore, some of the bounds stated in this thesis for space as defined next are off by a constant as compared to the cited sources.

- The *clause space* of π is $Sp(\pi) = \max_{\mathbb{C} \in \pi}\{Sp(\mathbb{C})\}$.
- The *variable space* of π is $VarSp(\pi) = \max_{\mathbb{C} \in \pi}\{VarSp(\mathbb{C})\}$.

We define the length of deriving a clause A from F as $L(F \vdash A) = \min_{\pi : F \vdash A}\{L(\pi)\}$, where the minimum is taken over all resolution derivations of A. The width $W(F \vdash A)$, clause space $Sp(F \vdash A)$, and variable space $VarSp(F \vdash A)$ of deriving A from F are defined completely analogously. The length, width, clause space and variable space of refuting F is $L(F \vdash 0)$, $W(F \vdash 0)$, $Sp(F \vdash 0)$, and $VarSp(F \vdash 0)$, respectively, where as before 0 denotes the contradictory empty clause.

In this thesis, we will be almost exclusively interested in the clause space of general resolution refutations. When we write simply "space" for brevity, we mean clause space.

As an aside, we note that if one wanted to be really precise, the size and space measures should probably measure the number of *bits* needed rather than the number of literals. However, counting literals makes matters substantially cleaner, and the difference is at most a logarithmic factor. Therefore, counting literals seems to be the established way of measuring formula size and variable space.

Using the "configuration-style" description of resolution in Definition 4.14, a tree-like resolution derivation can be defined as a derivation where a clause has to be erased as soon as it has been used in an inference step. Restricting the resolution derivations to tree-like resolution, we can define the minimum length $L_{\mathfrak{T}}(F \vdash 0)$, clause space $Sp_{\mathfrak{T}}(F \vdash 0)$, and variable space $VarSp_{\mathfrak{T}}(F \vdash 0)$ of refuting F in tree-like resolution in analogy with the measures in Definition 4.15. Note that the minimum width measures in general and tree-like resolution coincide, so it makes no sense to make a separate definition for $W_{\mathfrak{T}}(F \vdash 0)$.

For technical reasons, it is sometimes convenient to add a rule for *weakening* in resolution, saying that we can always derive a weaker clause $C' \supseteq C$ from C. It is easy to show that any weakening steps can always be eliminated from a resolution refutation without increasing the length, width or space.

Restrictions are another technical tool that we will use to to simplify some of the proofs.

Definition 4.16 (Restriction). A *partial assignment* or *restriction* ρ is a partial function $\rho : X \mapsto \{0, 1\}$, where X is a set of Boolean variables. We identify ρ with the set of literals $\{a_1, \ldots, a_m\}$ set to true by ρ. The *ρ-restriction* of a clause C is defined to be

$$C\restriction_{\rho} = \begin{cases} 1 & \text{(i.e., the trivially true clause) if } Lit\big(C\big) \cap \rho \neq \emptyset, \\ C \setminus \{\overline{a} \mid a \in \rho\} & \text{otherwise.} \end{cases}$$

This definition is extended to set of clauses by taking unions.

We write $\rho(\neg C)$ to denote the minimal restriction fixing C to false, i.e., $\rho(\neg C) = \{\overline{a} \mid a \in C\}$.

$\pi =$

1.	$x \vee z$	Axiom in F
2.	$\overline{z} \vee y$	Axiom in F
3.	$x \vee \overline{y} \vee u$	Axiom in F
4.	$\overline{y} \vee \overline{u}$	Axiom in F
5.	$u \vee v$	Axiom in F
6.	$\overline{x} \vee \overline{v}$	Axiom in F
7.	$\overline{u} \vee w$	Axiom in F
8.	$\overline{x} \vee \overline{u} \vee \overline{w}$	Axiom in F
9.	$x \vee y$	Res$(1,2)$
10.	$x \vee \overline{y}$	Res$(3,4)$
11.	$\overline{x} \vee u$	Res$(5,6)$
12.	$\overline{x} \vee \overline{u}$	Res$(7,8)$
13.	x	Res$(9,10)$
14.	$\overline{x}$	Res$(11,12)$
15.	0	Res$(13,14)$

(a) Resolution refutation π.

$\pi{\restriction}_x =$

1.	1	
2.	$\overline{z} \vee y$	Axiom in $F{\restriction}_x$
3.	1	
4.	$\overline{y} \vee \overline{u}$	Axiom in $F{\restriction}_x$
5.	$u \vee v$	Axiom in $F{\restriction}_x$
6.	$\overline{v}$	Axiom in $F{\restriction}_x$
7.	$\overline{u} \vee w$	Axiom in $F{\restriction}_x$
8.	$\overline{u} \vee \overline{w}$	Axiom in $F{\restriction}_x$
9.	1	
10.	1	
11.	u	Res$(5,6)$
12.	$\overline{u}$	Res$(7,8)$
13.	1	
14.	0	Res$(11,12)$
15.	0	

(b) Restriction $\pi{\restriction}_x$ setting x to true.

Figure 4.4: Proof by example that restrictions preserve resolution refutations.

Proposition 4.17. *If π is a resolution refutation of F and ρ is a restriction on $Vars(F)$, then $\pi{\restriction}_\rho$ can be transformed into a resolution refutation of $F{\restriction}_\rho$ in at most the same length, width and space as π.*

See Figure 4.4 for an illustration of this using our running example resolution refutation. In this case, the restriction results in a legal resolution refutation, but in general we might need the weakening rule to show that $\pi{\restriction}_\rho$ is a refutation of $F{\restriction}_\rho$. The formal proof is an easy induction over the derivations steps in π.

4.3 A Review of Some Results

It is not hard to show that any unsatisfiable CNF formula F over n variables is refutable in length $2^{n+1} - 1$, using the decision tree construction sketched in Figure 4.3. Also, the maximal refutation width is clearly at most the number of variables $n + 1$. Esteban and Torán [39] proved that the clause space of refuting F is upper-bounded by the formula size. More precisely, the minimal clause space is at most the number of clauses, or the number of variables, plus a small constant, or in formal notation $Sp(F \vdash 0) \leq \min\{|F|, |Vars(F)|\} + \mathrm{O}(1)$. Again, this follows by studying resolution refutations constructed as in Figure 4.3. The height of the decision tree is at most the number of variables, and it can be shown that any resolution refutation described by a binary tree of height at most h can be carried out in clause space $h + \mathrm{O}(1)$.

We will need the fact that there are polynomial-size families of k-CNF formulas that are very hard with respect to length, width and clause space, essentially meeting the upper bounds just stated.

Theorem 4.18 ([4, 13, 18, 21, 29, 79, 82]). *There are arbitrarily large unsatisfiable 3-CNF formulas F_n of size $\Theta(n)$ with $\Theta(n)$ clauses and $\Theta(n)$ variables for which it holds that $L(F_n \vdash 0) = \exp(\Theta(n))$, $W(F_n \vdash 0) = \Theta(n)$ and $Sp(F_n \vdash 0) = \Theta(n)$.*

Clearly, for such formulas F_n it must also hold that $\Omega(n) = \mathit{VarSp}(F_n \vdash 0) = \mathrm{O}(n^2)$. We note in passing that determining the exact variable space complexity of a formula family as in Theorem 4.18, or even proving a lower bound $\omega(n)$ on the variable space, was mentioned as an open problem in [4]. To the best of our knowledge, this problem is still unsolved.

If a resolution refutation has constant width, it is easy to see that it can be carried out in length polynomial in the number of variables (just count the maximum possible number of distinct clauses). Conversely, if all refutations of a formula are very wide, it seems reasonable that any refutation of this formula must be very long as well. This intuition was made precise by Ben-Sasson and Wigderson [21]. We state their theorem in the more explicit form of Segerlind [76].

Theorem 4.19 ([21]). *The width of refuting an unsatisfiable CNF formula F is bounded from above by*

$$W(F \vdash 0) \leq W(F) + 1 + 3\sqrt{n \ln L(F \vdash 0)} \ ,$$

where n is the number of variables in F.

Bonet and Galesi [25] showed that this bound on width in terms of length is essentially optimal. For the special case of tree-like resolution, however, it is possible get rid of the dependence of the number of variables and obtain a tighter bound.

Theorem 4.20 ([21]). *The width of refuting an unsatisfiable CNF formula F in tree-like resolution is bounded from above by $W(F \vdash 0) \leq W(F) + \log L_{\mathfrak{T}}(F \vdash 0)$.*

For reference, we collect the result in [25] together with some other bounds showing that there are formulas that are easy with respect to length but moderately hard with respect to width and clause space, and state them as a theorem.[3]

Theorem 4.21 ([4, 25, 78]). *There are arbitrarily large unsatisfiable 3-CNF formulas F_n of size $\Theta(n^3)$ with $\Theta(n^3)$ clauses and $\Theta(n^2)$ variables such that $W(F_n \vdash 0) = \Theta(n)$ and $Sp(F_n \vdash 0) = \Theta(n)$, but for which there are resolution refutations $\pi_n : F_n \vdash 0$ in length $L(\pi_n) = \mathrm{O}(n^3)$, width $W(\pi_n) = \mathrm{O}(n)$ and clause space $Sp(\pi_n) = \mathrm{O}(n)$.*

[3] Note that [25], where an explicit resolution refutation upper-bounding the proof complexity measures is presented, does not talk about clause space, but it is straightforward to verify that the refutation there can be carried out in length $\mathrm{O}(n^3)$ and clause space $\mathrm{O}(n)$.

As was mentioned in Chapter 2, the fact that all known lower bounds on refutation clause space coincided with lower bounds on width lead to the conjecture that the width measure is a lower bound for the clause space measure. This conjecture was proven true by Atserias and Dalmau [10].

Theorem 4.22 ([10]). *For any unsatisfiable CNF formula F, $Sp(F \vdash 0) - 3 \geq W(F \vdash 0) - W(F)$.*

In other words, the extra clause space exceeding the minimum 3 needed for any resolution refutation is bounded from below by the extra width exceeding the width of the formula.

An immediate corollary of Theorem 4.22 is that for polynomial-size k-CNF formulas, constant clause space implies polynomial proof length. We are interested in finding out what holds in the other direction, i.e., if upper bounds on length imply upper bounds on space. For the special case of tree-like resolution, it is known that there is an upper bound on clause space in terms of length exactly analogous to the one on width in terms of length in Theorem 4.20.

Theorem 4.23 ([39]). *The clause space of refuting an unsatisfiable CNF formula F in tree-like resolution is bounded from above by $Sp_{\mathfrak{T}}(F \vdash 0) \leq \lceil \log L_{\mathfrak{T}}(F \vdash 0) \rceil + 2$.*

For general resolution, since clause space is lower-bounded by width according to Theorem 4.22, the separation of width and length of [25] in Theorem 4.21 tells us that k-CNF formulas refutable in polynomial length can still have "somewhat spacious" minimum-space refutations. But exactly how spacious can they be? Does space behave as width with respect to length also in general resolution, or can one get stronger lower bounds on space for formulas refutable in polynomial length?

All polynomial lower bounds on clause space known prior to this thesis can be explained as immediate consequences of Theorem 4.22 applied on lower bounds on width. Clearly, any space lower bounds derived in this way cannot get us beyond the "Ben-Sasson–Wigderson barrier" implied by Theorem 4.19 saying that if the width of refuting F is $\omega\big(\sqrt{|F| \log|F|}\big)$, then the length of refuting F must be superpolynomial in $|F|$. Also, since matching upper bounds on clause space have been known for all of these formula families, they have not been candidates for showing stronger separations of space and length. Thus, the best known separation of clause space and length prior to this thesis was provided by the formulas in Theorem 4.21 refutable in linear length $L(F_n \vdash 0) = \mathrm{O}(|F_n|)$ but requiring space $Sp(F_n \vdash 0) = \Theta\big(\sqrt[3]{|F_n|}\big)$, as implied by the same bound on width.

Let us also discuss upper bounds on what kind of separations are a priori possible. Given any resolution refutation $\pi : F \vdash 0$, we can write down its DAG representation G_π (described on page 42) with $L(\pi)$ vertices corresponding to the clauses, and with all non-source vertices having fan-in 2. We can then transform π into as space-efficient a refutation as possible by considering an optimal black pebbling of G_π (soon to be formally defined in Definition 5.1) as follows: when a pebble is placed on a vertex we derive the corresponding clause, and when the pebble is removed again we erase the clause from memory. This yields a refutation π' in clause space $\mathsf{Peb}(G_\pi)$ (incidentally, this is the original definition in [39] of the clause space of a resolution refution π). Since it is known that any constant indegree DAG on n vertices can be black-pebbled in cost $\mathrm{O}(n/\log n)$ (see Theorem 5.4), this shows that $Sp(F \vdash 0) = \mathrm{O}\big(L(F \vdash 0)/\log L(F \vdash 0)\big)$ is an upper bound on space in terms of length.

Now we can rephrase the question above about space and length in the following way: Is there a Ben-Sasson–Wigderson kind of lower bound, say $L(F \vdash 0) = \exp\big(\Omega\big(Sp(F \vdash 0)^2/|F|\big)\big)$, on length in terms of space? Or do there exist k-CNF formulas F with short refutations but maximum possible refutation space $Sp(F \vdash 0) = \Omega\big(L(F \vdash 0)/\log L(F \vdash 0)\big)$ in terms of length? Note that the refutation length $L(F \vdash 0)$ must indeed be short in this case—essentially linear, since any formula F can be refuted in space $\mathrm{O}(|F|)$ as was noted above. Or is the relation between refutation space and refutation length somewhere in between these extremes?

This is the main question addressed in this thesis. We show that clause space and length can be strongly separated in the sense that there are formula families with maximum possible refutation clause space in terms of length. The same result also yields an almost optimal separation of clause space and width.

Theorem 4.24 (Corollary 2.6 restated). *For all $k \geq 6$ there is a family $\{F_n\}_{n=1}^{\infty}$ of k-CNF formulas of size $\Theta(n)$ that can be refuted in resolution in length $L(F_n \vdash 0) = \mathrm{O}(n)$ and width $W(F_n \vdash 0) = \mathrm{O}(1)$ but require clause space $Sp(F_n \vdash 0) = \Omega(n/\log n)$.*

Bibliography

[1] Douglas Adams. *The Restaurant at the End of the Universe*. Pan Macmillan, 1980.

[2] Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *Journal of Combinatorial Theory*, 43:196–204, 1986.

[3] Michael Alekhnovich. Lower bounds for k-DNF resolution on random 3-CNFs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05)*, pages 251–256, May 2005.

[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] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 448–456, May 2002.

[6] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 210–219, October 2001.

[7] Noga Alon and Michael Capalbo. Smaller explicit superconcentrators. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03)*, pages 340–346, 2003.

[8] Albert Atserias and Maria Luisa Bonet. On the automatizability of resolution and related propositional proof systems. *Information and Computation*, 189 (2):182–201, March 2004.

[9] Albert Atserias, Maria Luisa Bonet, and Juan Luis Esteban. Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. *Information and Computation*, 176(2):136–152, August 2002.

[10] Albert Atserias and Victor Dalmau. A combinatorical characterization of resolution width. In *Proceedings of the 18th IEEE Annual Conference on Computational Complexity (CCC '03)*, pages 239–247, July 2003. Journal version to appear in *Journal of Computer and System Sciences*.

[11] Sven Baumer, Juan Luis Esteban, and Jacobo Torán. Minimally unsatisfiable CNF formulas. *Bulletin of the European Association for Theoretical Computer Science*, 74:190–192, June 2001.

[12] Paul Beame. Proof complexity. In Steven Rudich and Avi Wigderson, editors, *Computational Complexity Theory*, volume 10 of *IAS/Park City Mathematics Series*, pages 199–246. American Mathematical Society, 2004.

[13] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31 (4):1048–1075, 2002.

[14] Paul Beame, Henry Kautz, and Ashish Sabharwal. Understanding the power of clause learning. In *Proceedings of the 18th International Joint Conference in Artificial Intelligence (IJCAI '03)*, pages 94–99, 2003.

[15] Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present, and future. *Bulletin of the European Association for Theoretical Computer Science*, 65:66–89, June 1998.

[16] 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.

[17] Eli Ben-Sasson. Personal communication, 2007.

[18] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, August 2003.

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

[20] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. Submitted, April 2008.

[21] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001.

[22] Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

[23] Maria Luisa Bonet, Carlos Domingo, Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi. Non-automatizability of bounded-depth frege proofs. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC '99)*, pages 15–23, May 1999.

[24] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000.

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

[26] Josh Buresh-Oppenheim and Toniann Pitassi. The complexity of resolution refinements. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 138–147, June 2003.

[27] Samuel R. Buss. Some remarks on the lengths of propositional proofs. *Archive for Mathematical Logic*, 34:377–394, 1995.

[28] Samuel R. Buss, editor. *Handbook of Proof Theory.* Elsevier Science, Amsterdam, 1998.

[29] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[30] Peter Clote and Evangelos Kranakis. *Boolean Functions and Computation Models.* Springer, 2002.

[31] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

[32] Stephen A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9:308–316, 1974.

[33] Stephen A. Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979.

[34] Stephen A. Cook and Ravi Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.

[35] Dirk van Dalen. *Logic and Structure.* Universitext. Springer, Berlin, 3rd, augmented edition, 1994.

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

[37] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[38] Juan Luis Esteban, Nicola Galesi, and Jochen Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321 (2-3):347–370, August 2004.

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

[40] Juan Luis Esteban and Jacobo Torán. A combinatorial characterization of treelike resolution space. *Information Processing Letters*, 87(6):295–300, 2003.

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

[42] John R. Gilbert and Robert Endre Tarjan. Variations of a pebble game on graphs. Technical Report STAN-CS-78-661, Stanford University, 1978. Available at `http://infolab.stanford.edu/TR/CS-TR-78-661.html`.

[43] James Gleick. A bug and a crash. *New York Times Magazine*, December 1996. Available at `http://www.around.com/`.

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

[45] Philipp Hertel and Toniann Pitassi. Exponential time/space speedups for resolution and the PSPACE-completeness of black-white pebbling. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pages 137–149, October 2007.

[46] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, April 1977.

[47] Balasubramanian Kalyanasundaram and George Schnitger. On the power of white pebbles. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 258–266, 1988.

[48] Henry Kautz and Bart Selman. The state of SAT. *Discrete Applied Mathematics*, 155(12):1514–1524, June 2007.

[49] Maria M. Klawe. A tight bound for black and white pebbles on the pyramid. *Journal of the ACM*, 32(1):218–228, January 1985.

[50] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-3):123–140, 2001.

[51] Jan Krajíček, Pavel Pudlák, and Alan R. Woods. An exponential lower bound to the size of bounded depth frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1):15–40, 1995.

[52] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[53] Oliver Kullmann. An application of matroid theory to the SAT problem. In *Proceedings of the 15th Annual IEEE Conference on Computational Complexity (CCC '00)*, pages 116–124, July 2000.

[54] Thomas Lengauer and Robert Endre Tarjan. Upper and lower bounds on time-space tradeoffs. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC '79)*, pages 262–277, May 1979.

[55] Thomas Lengauer and Robert Endre Tarjan. The space complexity of pebble games on trees. *Information Processing Letters*, 10(4/5):184–188, July 1980.

[56] Friedhelm Meyer auf der Heide. A comparison of two variations of a pebble game on graphs. *Theoretical Computer Science*, 13(3):315–322, 1981.

[57] The Millennium Problems of the Clay Mathematics Institute, May 2000. See `http://www.claymath.org/millennium/`.

[58] Cleve Moler. A tale of two numbers, 1995. Available at `http://www.mathworks.com/company/newsletters/news_notes/clevescorner/`.

[59] Jakob Nordström. Stålmarck's method versus resolution: a comparative theoretical study. Master's thesis, Stockholm University, 2001. Available at `http://www.csc.kth.se/~jakobn/research/`.

[60] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution (Extended abstract). In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, pages 507–516, May 2006.

[61] Jakob Nordström. A simplified way of proving trade-off results for resolution. Technical Report TR07-114, Electronic Colloquium on Computational Complexity (ECCC), September 2007.

[62] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM Journal on Computing*, 2008. To appear. Preliminary version available at `http://www.csc.kth.se/~jakobn/research/`.

[63] Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution (Extended abstract). In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, May 2008. To appear.

[64] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[65] Christos H. Papadimitriou and David Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*, 37(1):2–13, 1988.

[66] Joachim Parrow. Programmeraren som schaman. *Forskning & Framsteg*, 2:14–19, February 1998. In Swedish. Available at `http://fof.se/?id=98214`.

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

[68] Nicholas Pippenger. Pebbling. Technical Report RC8258, IBM Watson Research Center, 1980. Appeared in Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science, Japan.

[69] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993.

[70] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, March 1999.

[71] Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. Manuscript. Available at the webpage `http://www.mi.ras.ru/~razborov/`, July 2003.

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

[73] Ashish Sabharwal. *Algorithmic Applications of Propositional Proof Complexity*. PhD thesis, University of Washington, Seattle, 2005.

[74] Ashish Sabharwal, Paul Beame, and Henry Kautz. Using problem structure for efficient clause learning. In *6th International Conference on Theory and Applications of Satisfiability Testing (SAT '03), Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2004.

[75] The international SAT Competitions. `http://www.satcompetition.org`.

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

[77] Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004.

[78] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33(3):277–280, May 1996.

[79] 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.

[80] Jacobo Torán. Space and width in propositional resolution. *Bulletin of the European Association for Theoretical Computer Science*, 83:86–104, June 2004.

[81] Grigori Tseitin. On the complexity of derivation in propositional calculus. In A. O. Silenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, New York-London, 1968.

[82] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1): 209–219, January 1987.

[83] Alasdair Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1(4):425–467, 1995.