On Resolution Complexity of Matching Principles

Stefan Dantchev

BRICS Dissertation Series DS-02-2

ISSN 1396-7002 May 2002

On Resolution Complexity of Matching Principles

Stefan Dantchev

PhD Dissertation

BRICS
Department of Computer Science
University of Aarhus
Denmark

On Resolution Complexity of Matching Principles

A Dissertation
Presented to the Faculty of Science
of the University of Aarhus
in Partial Fulfilment of the Requirements for the
PhD Degree

by
Stefan Dantchev
April 2, 2002

To my parents,
Maria and Stoyan

Abstract

Studying the complexity of mathematical proofs is important not only for automated theorem proving, but also for Mathematics as a whole. Each significant result in this direction would potentially have a great impact on Foundations of mathematics.

Surprisingly enough, the general Proof Complexity is closely related to Propositional Proof Complexity. The latter area was founded by Cook and Reckhow in 1979, and enjoyed quite a fast development since then. One of the main research directions is finding the precise complexity of some natural combinatorial principle within a relatively weak propositional proof system.

The results in the thesis fall in this category. We study the Resolution complexity of some Matching Principles. The three major contributions of the thesis are as follows.

Firstly, we develop a general technique of proving resolution lower bounds for the perfect matching principles based on regular planar graphs. No lower bounds for these were known prior to our work. As a matter of fact, one such problem, the Mutilated Chessboard, was suggested as hard to automated theorem provers in 1964, and remained open since then. Our technique proves a tight resolution lower bound for the Mutilated Chessboard as well as for Tseitin tautologies based on rectangular grid graph. We reduce these problems to Tiling games, a concept introduced by us, which may be of interest on its own.

Secondly, we find the exact Tree-Resolution complexity of the Weak Pigeon-Hole Principle. It is the most studied combinatorial principle, but even its Tree-Resolution complexity was unknown prior to our work. We develop a new, more general method for proving Tree-Resolution lower bounds. We also define and prove non-trivial upper bounds on worst-case proofs of the Weak Pigeon-Hole Principle. The worst-case proofs are first introduced by us, as a concept opposite to the optimal proofs.

Thirdly, we prove Resolution width-size trade-offs for the Pigeon-Hole Principle. Proving the size lower bounds via the width lower bounds was known since the seminal paper of Haken, who first proved an exponential lower bound for the ordinary Pigeon-Hole Principle. The width-size trade-offs however were not studied at all prior to our work. Our result gives an optimal width-size trade-off for resolution in general.

Acknowledgements

I am grateful to the Danish National Research Foundation for funding my PhD research at BRICS. I would like to thank all the BRICS people for making the place so friendly and scientifically stimulating. I would like to specially mention the people who contributed most to BRICS development and current status: Uffe Engberg, Mogens Nielsen, Erik Meineche Schmidt, and Glynn Winskel.

I would like to thank my advisors, Søren Riis and Erik Meineche Schmidt. My deepest thanks go to Søren, my scientific advisor, with whom I worked and achieved most of the results in the present thesis. Without him, not only would I not have been able to achieve results of the same strength, but also I would not even know that the fascinating area of Propositional Proof Complexity exists. Many thanks to Erik for giving me support and encouragement, whenever I needed it.

I wish to thank my MSci advisor Valentin Brimkov with whom I worked on some different problems during my PhD studies. He helped me by proof-reading some chapters of the thesis, too.

I am thankful to all the BRICS lecturers and guests whose courses I attended and benefited from. These include Gerth Stølting Brodal, Ivan Damgård, Devdatt Dubhashi, Ulrich Kohlenbach, Kevin Compton, Peter Bro Miltersen, Aleksandar Pekec, Vinodchandran Variyam and many others.

I wish to thank the secretaries, Janne Christensen and Karen Møller. They helped me with all the practical things, and made my life in Denmark as easy as possible.

Many thanks to the fellow students, Federico Crazzolara, Frank Valencia, Mario José Cáccamo, Oliver Möller, and Riko Jacob. We started our PhD studies together in September, 1998, and have had a good time ever since.

I am grateful to the Department of Computer Science of Queen Mary college, University of London for allowing me to stay there for more than an year, when Søren moved permanently there.

I would like to thank Jan Krajicek, Pavel Pudlak and Jiri Sgall for their hospitality during my short stay in Mathematical Institute of the Academy of Sciences of the Czech Republic in Prague. I have had many interesting and helpful discussion with them.

Finally, I would like to thank anybody who (thinks that he or she) deserves it, but whom I have forgotten to mention in the above list of people.

Stefan Dantchev,
London, April 2, 2002.

Contents

Chapter 1

Introduction

1.1 Introduction

Given a mathematical theorem, what is the size of its shortest proof within an appropriate formalisation of Set Theory, say, ZF? What is the minimal knowledge, we need, in order to prove a particular mathematical theorem, i.e. what is the weakest formal system which allows us to prove the theorem? Given a formal system, and different tautologies, can we say whether one of them is "harder" than the other within this system? Finally, considering only the class of theorems which have "short" proofs within a particular system, is it always possible to find such, or at least not much bigger, proof? If not, for how weak systems is it possible?

It is clear that (even a partial) answer to one of these questions would have a great impact on the foundations of Mathematics. This motivates the development of areas of mathematics such as Proof Theory and, more recently, Proof Complexity. In order to justify the research in the latter area, we will start with

Some history

The idea of considering the lengths of a mathematical proofs as well as efficient algorithms finding a short proof provided it exists can be tracked back as early as 1956. The famous Gödel's letter to von Neumann is probably the first written document, where this ideas appeared explicitly, in a form allowing their formalisation. Gödel writes:

"...It is evident that one can easily construct a Turing machine which, for each formula F of the predicate calculus and for every natural number n, will allow one to decide if F has a proof of length n [length = number of symbols]. Let $\Psi(F, n)$ be the number of steps that the machine requires for that and let $\varphi(n) = \max_F \Psi(F, n)$. The question is, how fast does $\varphi(n)$ grow for an optimal machine. One can show that $\varphi(n) \geq Kn$. If there actually were a machine with $\varphi(n) \sim Kn$ (or even only with $\varphi(n) \sim Kn^2$), this would have consequences of the greatest magnitude. That is to say, it would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely **[Gödel's footnote:** apart from the postulation of axioms**]** replaced by machines. One would indeed have to simply select an n so large that, if the machine yields no result, there would then also be no reason to think further about the problem. Now, it seems to me, however, to be totally within the realm of possibility that $\varphi(n)$ grows slowly. For 1.) it seems that $\varphi(n) \geq Kn$ is the only estimate that can be derived from a generalisation of the proof of the Entscheidungsproblem; 2.) $\varphi(n) \sim Kn$ (or $\sim Kn^2$) means, of course, simply that the number of steps vis-à-vis *dem blossen Probieren* **[brute force]** can be reduced from N to $\log N$ (or $(\log N)^2$) **[most likely $N = 2^n$ here]**. Such strong reductions do indeed occur, however, in the case of other finite problems, e.g., in the case of calculating a quadratic residue by means of repeated application of the law of reciprocity. It would be interesting to know, for example, what the situation is in the case of determining whether a number is a prime number, and in the case of finite combinatorial problems, how strongly in general the number of steps vis-à-vis the blossen Probieren can be reduced..." (This is the translation of the letter which appears in [50]).

We have included such a long part of the letter in order to be able to make the following very important points, especially because the letter is often quoted in a somewhat negative context (see the second of the our remarks):

1. Gödel was probably the first man to understand the importance of the **automatisability** of (very strong) **proof systems**. He put the question in a rigorous context, and pointed out that it is very important not only to some relatively small areas of mathematics, such as combinatorial optimisation, but also for mathematics as a whole.

2. Gödel **did not suggest** that $\mathcal{NP} = \mathcal{P}$ (or even $\mathcal{L}$, linear time) as it may seem, and as many people in structural complexity community erroneously claim. A more careful reader would realise that the only thing Gödel said is that he **does not see any way for proving a better than linear** (quadratic) lower bound for an $\mathcal{NP}$-language. As a matter of fact, this is still the **best lower bound** we know **up to date**.

Surprisingly enough, the very general consideration, which appear in the letter are strongly connected to concepts and notions which, at first glance, seem to be much weaker. These are the very basic and fundamental complexity classes as defined by the founders of the modern Computational (Structural) Complexity Theory, Cook and Levin. In [19] and [34] they independently gave the rigorous definitions of the complexity classes $\mathcal{P}$ and $\mathcal{NP}$, and posed the $\mathcal{P}$ vs $\mathcal{NP}$ question. It was not just a coincidence that Cook's paper is entitled "The complexity of theorem proving procedures", and Levin mentioned the problem of "... the search for proofs of finite length..." among the most important so-called exhaustive-search problems.

Later on, Cook, together with Reckhow [20], founded the modern Propositional Proof Complexity. Event though their motivation was the $\mathcal{NP}$ vs $co\!-\!\mathcal{NP}$ question, their work can be considered as an important step towards understanding the complexity of mathematical proofs in general.

The rest of the thesis

is organised as follows.

In the section 1.2 of this chapter, we first give the very fundamental definitions and results of Propositional Proof Complexity as they appear in the original paper of Cook and Reckhow [20]. We then describe the main research directions as they have developed in the following two, almost three, decades, up to now.

The section 1.3 is an informal overview of results and open problems in Propositional Proof Complexity. It contains description of some well known and well studied propositional proof system as well as families of tautologies. Many interesting results and open problems are listed. The section is not meant to be a comprehensive survey of the area. The aim is rather to give the flavour of Proof Complexity to a reader who does not know anything about it. For a good, now a bit outdated, survey of results (without proofs) and open problems we suggest [8].

In section 1.4, we state the results achieved by us. We first survey the known result about resolution complexity of matching principles, and then explain what the contributions of the present thesis are.

The next three chapters contain our results. They are extended and, hopefully, improved version of the papers [23], [24] and [22]. These chapters are therefore self-contained, and can be read separately.

1.2 Propositional Proof Complexity vs Computational Complexity

Fundamental Definitions and Results

We let $TAUT$ denote the set of tautologies over any adequate set of connectives. The famous $\mathcal{NP}$ vs $co-\mathcal{NP}$ can be stated as the following proposition.

Proposition 1.1 *$\mathcal{NP}$ is closed under complementation if and only if $TAUT$ is in $\mathcal{NP}$*

In order to study the complexity of proofs one needs the formal definition of a proof system

Definition 1.2 *If $L \subseteq \Sigma^*$, a **proof system** for L is a* polynomial-time computable onto function $f : \Sigma_1^* \to L$ *for some alphabet Σ_1. The proof system is **polynomially bounded** iff there is a* polynomial $p(n)$ *such that for all $y \in L$ there is $x \in \Sigma_1^*$ such that $y = f(x)$ and $|x| \leq p(|y|)$. Here $|.|$denotes the size.*

As Cook and Reckhow pointed out the above definition not only includes all the "natural" proof systems, but also allows to define (somewhat artificial) new systems, specific to a $co-\mathcal{NP}$ language L. Clearly, if there is polynomially bounded proof system for a $co-\mathcal{NP}$-complete language, the $co-\mathcal{NP} = \mathcal{NP}$.

$\mathcal{NP}$, itself, can be defined in terms of proof systems as follows.

Proposition 1.3 *A set L is in $\mathcal{NP}$ iff $L = \emptyset$ or L has a polynomially bounded proof system.*

In order to compare the strength of the proof systems, we need the notion of *p(olynomial)-simulation.*

Definition 1.4 *If $f_1 : \Sigma_1^* \to L$ and $f_2 : \Sigma_2^* \to L$ are proof systems for L, then f_2 p-simulates f_1 provided there is a polynomial-time computable function $g : \Sigma_1^* \to \Sigma_2^*$ such that $f_2 (g (x)) = f_1 (x)$ for all $x \in \Sigma_1^*$.*

It is an important open problem whether there exists an *optimal proof system* for $co - \mathcal{NP}$, i.e. a proof system that p-simulates any other.

Main Research Directions

There are the following main streams in the Propositional Proof Complexity

1. To prove lower bounds for stronger and stronger systems. The best up-to-date result is that there are "hard" tautologies for Bounded-depth Frege with counting axioms (see the next chapter for the relevant definitions). Lower bounds for Frege systems, however, seem to be far beyond the current techniques. We cannot even rule out the possibility that Frege is an optimal proof system. The importance of this stream comes from the fact, that the development of techniques, used to prove super-polynomial lower bounds for stronger and stronger systems, could eventually lead to a lower bound that applies to any proof systems for a $co - \mathcal{NP}$, thus proving $co - \mathcal{NP} \neq \mathcal{NP}$.

2. To study the precise limitations of the weaker propositional proof systems. There are many such systems that are "natural", in the sense that they are used in the real life, as a basis for SAT-solving algorithms. Therefore it is interesting to know the strength of a particular proof system for both theoretical and practical point of view. The results from the thesis fall in this stream as we consider the resolution complexity of some natural combinatorial principles, namely the matching principles.

3. To show (non)automatisability results. The positive results in this direction are very weak. Introducing weak notions of automatisability, we can prove such results, but only for systems as weak as tree-like resolution. On the other hand, if we adapt a natural definition of automatisability, to find a proof of size only polynomially bigger than the optimal, even tree-like resolution is not automatisable under some sensible complexity assumptions.

1.3 Propositional proof systems and (families of) tautologies

This section is an informal overview of results and open problems in Propositional Proof Complexity. It contains description of some well known and well studied propositional proof system as well as families of tautologies. The section is self-contained, and we believe it would be of interest of a person who wants to get the favour of the are.

There are two kind of objects in Propositional Proof Complexity: **Propositional Proof Systems** and **Tautologies**. Ideally, given a particular proof system and particular tautology, we would like to know what the size is of the shortest proof of the tautology within the system.

We first give the basic definitions, denotations and conventions that apply to all the proof systems and tautologies. We then list some natural and well studied proof systems as well as families of tautologies. Finally, we give a partial map of the proof systems considered in this section.

Some terminology, denotations, and conventions

We shall first introduce some basic concepts.

A (finite) set of **variables** is given. Usually, we consider *propositional variables*, having the value either true or false. Sometimes, we also consider, and use the adjective "propositional", for $0-1$ variables, interpreting 0 as false and 1 as true. A **formula** is built upon variables. Apart from the usual, propositional, formulae, we shall call "formula" any *polynomial equation/inequality* built upon some $0-1$ variables. An important special form of formulae are clauses. A **clause** is a *disjunction* of literals, each **literal** being either a propositional variable or a negation of such a variable. It is straightforward to encode a clause as a linear inequality or a polynomial equation. The clause $l_1 \vee l_2 \vee \ldots \vee l_w$ is equivalent to the inequality

$$T(l_1) + T(l_2) + \ldots + T(l_w) \geq 1,$$

where

$$T(l_i) = \begin{cases} v_i & l_i \text{ is the variable } v_i \\ 1 - v_i & l_i \text{ is the negation of } v_i,\ \neg v_i \end{cases},$$

or the equation

$$M(l_1) \cdot M(l_2) \ldots M(l_w) = 0,$$

where

$$M(l_i) = \begin{cases} 1 - v_i & l_i \text{ is the variable } v_i \\ v_i & l_i \text{ is the negation of } v_i,\ \neg v_i \end{cases}.$$

A **propositional proof systems** operates on an *unsatisfiable* sets of formulae. There are **inference rules**, which are used derive new formulae from the already derived ones. Any such rule has to be **sound**, i.e. every truth assignment falsifying the conclusion has to falsify at least one of the premises. The initial set of formulae is *refuted*, when a contradiction is derived. The contradiction means usually the constant "false". However it may be encoded in several different ways, the empty clause, the inequality "$1 < 0$", the equality "$1 = 0$" etc. We only consider **complete** proof system, i.e. it is always possible to derive the contradiction from an unsatisfiable set of formulae. Apart from inference rules, a proof system has a set of axioms. The **axioms** are *tautologies*, i.e. true under any assignment, and are depend on the proof system only. As examples of axioms, we can mention "$v_j \leq 1$" and "$0 \leq v_j$", where v_j is a $0-1$ propositional variable, for inequalities-based systems; "$v_j - v_j^2 = 0$ for equalities-based systems; systems without axioms, such as resolution; or systems, such as Frege, having infinite set of axioms such as "$A \rightarrow (B \rightarrow A)$" for any formulae A, B.

We can now explain the general problem setting in **propositional proof complexity**. We are given a *proposition*, we would like to *prove* within a particular *proof system*. We first encode the *negation* of the proposition as a set of formulae, and then use the *inference rules* and *axioms* of the proof system in order to derive a *contradiction*.

By convention, we shall use "tautology" not only in the usual meaning, i.e. for a formula that evaluates to true under any assignment, but also for the negation of such a formula. We shall use "prove something" but what it really means is that we refute its negation.

So far, we have explained only the first word of the concept "proof complexity". We introduce several *complexity measures* of propositional proofs. In order to be able to speak about complexity, whenever we say "a tautology", we really mean a family of tautologies, parametrised by some integer n. Each measure we then use is a function in n. The most important one is the **size**, measured in the usual way, as the number of the symbols in the proof. An important exception are equalities-based proof systems, for which the size does not make much sense. The most important complexity measure there is the **degree**, that is the maximal degree over all the polynomials contained in the proof. A similar concept, specific to resolution only, is the **width**, i.e. the maximal number of literals, contained in a single clause, taken over all the clauses in the proof. In speaking about these measures, we shall use "big enough", "hard" etc. for the **exponential size** or **linear degree**; for **polynomial size** or **constant degree** we shall use words like "small", "easy" etc. Thus by saying that a certain tautology is hard for some proof system, we mean, that a family of tautologies, parametrised by some integer number n requires $\exp(\Omega(n))$ size proof or $\Omega(n)$ degree proof within the proof system. Another measure is the **space** of a proof, which will be defined shortly.

Another important concept is the *graph* of a proof. The vertices of the graph are all the formulae appearing in the proof. There is an edge, whenever the target of it has been derived, using the source as a premise. Clearly the graph is a *directed acyclic graph* (DAG). Thus each vertex is of bounded (constant) fan-in, as inference rules have constant size, and therefore constant number of premises. The fan-out may be unbounded as a formula can be used many times as a premise in deriving new formulae. The axioms and the formulae, representing the negation of the original proposition, are exactly the sources of the graph. The only sink is the contradiction. We can now define the **space** of a proof as the *pebbling number* of its graph.

We can also explain the connection between proof systems and (restricted) **models of computations**. Assuming an unsatisfiable set of clauses, we consider the following **search problem**: Given a truth assignment, find a formula from the given set, falsified under the assignment. We can take a *refutation* of the initial set of formulae, and transform it into an *algorithm*, solving the search problem as follows. We first turn around all the edges of the graph of the proof. The contradiction now becomes the only root (source) of the new graph, and the axioms and the initial formulae become the leaves (sinks). We perform a search in the new graph, starting from the root, which is falsified by any assignment, and always going to a vertex which is falsified under the given assignment. Such a vertex always exists as all the inference rules are *sound.* We end up at a leaf, which cannot be an axiom, as an axiom always evaluates to true, no matter what the assignment is. Therefore, we ended up in some of the initial formulae, and moreover it is false under the given assignment. The model of computations we get from the above procedure is a **restricted branching program**. The restrictions are the formulae, we are allowed to keep at a node and to query at an edge of the program. Clearly these restrictions are imposed by the inference rules of the proof system.

The last, but probably the most important, consequence of these consideration is so-called **Prover-Adversary game**. We shall give here only the general idea. The technique is used to prove lower bounds on the size of the proofs of (families of) propositions within a particular proof system. There are two players, Prover and Adversary. Adversary claims that the proposition is satisfiable. Prover's task is to convict him in lying. In doing so, we can assume that her strategy is kept as a restricted branching program, solving the search problem. A *position* in the game is a formula. Given a position, Prover queries another formula, and, depending on Adversary's answer and rules of the game, i.e. inference rules of the proof system, switches to another position. The game is over, and Adversary loses, when a position is reached, which is the negation of some of the original formulae or axioms. As Prover can always win, Adversary's task is to force her to memorise as much as possible. Therefore, any randomised Adversary's strategy which enforces big enough, subgraph in Prover's restricted branching program would prove the corresponding lower bound on the propositional proof.

Propositional proof systems

Resolution and restrictions, Tree-like and Regular resolution. Resolution is probably the simplest system we can think of. It was introduced as a proof system in [26], and as an algorithm for checking satisfiability in [25]. Resolution operates on a set of clauses. There are no additional axioms. The only rule is the following:

$$\frac{A \bigcup \{v_j\} \quad B\,\{\neg v_j\}}{A \bigcup B},$$

i.e. given two clauses and a variable, so that one of the clauses contains the variable, and the other contains the negated variable, we can derive a clause which is the union of the initial ones with the occurrences of the resolved variable removed.

There are two important restrictions of resolution: **Regular resolution**, if each variable is resolved *at most once* along any path in the refutation graph, and **Tree-like resolution**, if the refutation graph is a tree. These two restriction correspond to *read-once branching program* and *decision tree computational models,* respectively. Unrestricted resolution is, however, weaker than (general) branching program. Strange enough, the most successful SAT solvers are based on DLL procedure [25], which is in fact equivalent to Tree-like Resolution, one of the weakest propositional proof system. This can be explained by the intuitively obvious trade-off between strength and automatisability, which in its turn is strongly related to the design of good algorithmic heuristics.

Algebraic proof systems, Polynomial calculus and Nullstellensatz.

These systems are introduced in [18] and [4], respectively. **Polynomial calculus** operates on set of polynomial equations. There are three inference rules,

$$\frac{p(v) \quad q(v)}{p(v)+q(v)} \qquad \frac{p(v)}{\gamma \cdot p(v)} \qquad \frac{p(v)}{v_j \cdot p(v)},$$

which allows to derive any linear combination of some previously derived polynomials, and to weaken an already derived polynomial by multiplying it by some polynomial. There are the axioms

$$v_j - v_j^2$$

for each variable. The contradiction is represented by any constant, different from zero. Clearly, polynomial calculus p-simulates resolution. As a computational model, it is equivalent to *Groebner basis algorithms.*

Polynomials Q_i, satisfying

$$\sum_{i=1}^{m} Q_i(v) \cdot p(v) + \sum_{j=1}^{n} Q_{j+m}(v) \cdot \left(v_j - v_j^2\right) = 1,$$

can be extracted from a given polynomial-calculus proof. They form a **Nullstellensatz** proof, which is considered as a "static" version of the polynomial-calculus one. Note that, in general, the degree of the proof increases. There are indeed tautologies, having a small (constant) degree polynomial calculus proof, but require big (non-constant) degree Nullstellensatz proofs.

Inequalities based prof systems, Cutting planes and Lovasz-Schrijver.

Cutting planes was firstly known as an algorithm for Integer Programming, introduced in [30]. As a proof system it was first considered in [21]. Cutting planes operates on *linear inequalities*. There are rules, allowing to derive a positive linear combination of some previously derived inequalities or axioms:

$$\frac{a^T v \le \alpha \quad b^T v \le \beta}{(a+b)^T v \le (\alpha+\beta)} \qquad \frac{a^T v \le \alpha}{(\gamma \cdot a)^T v \le (\gamma \cdot \alpha)},$$

as well as the cutting rule,

$$\frac{(\gamma \cdot a)^T v \le \alpha}{a^T v \le \left\lfloor \frac{\alpha}{\gamma} \right\rfloor},$$

where $\gamma > 0$. There are the axioms

$$-v_j \le 0 \quad v_j \le 1 \tag{1.1}$$

for each variable. Clearly, Cutting planes p-simulate resolution, even without using the cutting rule.

Lovasz-Schrijver proof system is introduced in [35] both as a proof system and as an algorithm for relaxation of integer programs. It operates on *quadratic inequalities.* We can derive any positive linear combination of some already derived *quadratic inequalities* by using the rules

$$\frac{p(v) \le 0 \quad q(v) \le 0}{p(v) + q(v) \le 0} \qquad \frac{q(v) \le 0}{\gamma \cdot q(v) \le 0},$$

where $\gamma > 0$. We can also multiply a *linear inequality* by either a variable or a negation of it:

$$\frac{l(v) \le 0}{v_j \cdot l(v) \le 0} \qquad \frac{l(v) \le 0}{(1 - v_j) \cdot l(v) \le 0}.$$

The axioms are (1.1) as well as the new axioms

$$v_j - v_j^2 \le 0 \quad v_j^2 - v_j \le 0.$$

Again, it is easy to show that Lovasz-Schrijver p-simulates resolution.

It is worth mentioning that the only known lower bounds for these systems are obtained via a method called "effective interpolation" (see [39]). It would be extremely interesting to show lower bounds for some natural, i.e. combinatorial, tautologies.

Frege and Bounded-depth Frege

Frege proof system operates formulae. There are many variations in inference rules and axioms, but they all are essentially equivalent, i.e. p-simulates each other. Below we give a particular version of Frege systems, described in [13]. The only inference rule is *modus ponens*,

$$\frac{A \quad A \to B}{B},$$

and there are ten axioms:

$$\begin{array}{ll}
(A \wedge B) \to A & (A \wedge B) \to B \\
A \to (A \vee B) & B \to (A \vee B) \\
(A \to B) \to ((A \to \neg B) \to \neg A) & (\neg\neg A) \to A \\
A \to (B \to A \wedge B) & (A \to C) \to ((B \to C) \to (A \vee B \to C)) \\
A \to (B \to A) & (A \to B) \to (A \to (B \to C)) \to (A \to C)
\end{array}$$

Here A, B and C are not particular instances, but any formulae built from the variables v_j.

A restricted version of Frege systems is **Bounded-depth Frege** systems. There all the formulae in the proof are of *constant depth*, considered as AC_0 *boolean circuits* with and, or and not gates. The importance of these comes from the fact, that a proof of certain fragments of Bounded Arithmetic can be translated into a Bounded-depth Frege propositional proof (see [33] for details). Therefore, a super-polynomial lower bound on Bounded-depth Frege proofs of a certain principle implies the unprovability of the corresponding statement within the fragment of Bounded Arithmetic.

Tautologies

(Different versions of) the Pigeon-Hole Principle It is probably the most widely used combinatorial principle. There are m pigeons and n holes, $m > n$. The *Pigeon-Hole Principle* then states that there is no mapping from the pigeons to the holes, such that every pigeon goes to some hole(s) and no two pigeons go to the same hole. To encode it, we introduce propositional variables p_{ij}, $i \in [m]$, $j \in [n]$, with the obvious meaning, p_{ij} is true if and only if the i-th pigeon goes to the j-th hole. We consider several versions of the Pigeon-Hole Principle.

The *strongest* version, called simply the **Pigeon-Hole Principle**, and denoted by PHP_n^m. We allow the mapping to be a *relation*, i.e. a pigeon can go to *more than one hole.* PHP_n^m is encoded by the following set of clauses:

$$p_{i1} \vee p_{i2} \vee \ldots \vee p_{in} \quad \text{for } i \in [m] \tag{1.2}$$

$$\neg p_{ik} \vee \neg p_{jk} \quad \text{for } k \in [n]\,,\ i,j \in [m]\,,\ i \neq j. \tag{1.3}$$

The corresponding linear inequalities are $p_{i1}+p_{i2}+\ldots+p_{in} \geq 1$and $p_{ik}+p_{jk} \leq 1$, respectively.

Another version is so called **Functional Pigeon-Hole Principle**, $FPHP_n^m$, where a pigeon cannot split into many holes. To encode it, we add to the encoding of PHP_n^m the clauses

$$\neg p_{ij} \vee \neg p_{ik} \quad \text{for } i \in [m]\,,\ i,k \in [n]\,,\ j \neq k. \tag{1.4}$$

The functional pigeon-hole principle can be encoded as a system of *low-degree polynomial equations.* The set of equations $p_{i1}+p_{i2}+\ldots+p_{in} = 1$ and $p_{ij}p_{ik} = 0$ are equivalent to the set of clauses (1.2) and (1.4). Note that the first version, PHP_n^m, cannot be encoded in such a way *without extension variables.*

The **Bijective Pigeon-Hole Principle** is the last version we consider. It is the functional version, where the function is required to be a *bijection.* To encode it we add the set of clauses

$$p_{1j} \vee p_{2j} \vee \ldots \vee p_{mj} \quad \text{for } j \in [n]\,,$$

which together with (1.3) are equivalent to the set of equations $p_{1j}+p_{2j}+\ldots+p_{mj} = 1$ and $p_{ik}p_{jk} = 0$.

The Pigeon-Hole Principle has the distinction of the most studied combinatorial principle within the Propositional Proof Complexity framework. A short and most likely incomplete account follows.

The *ordinary* version, i.e. PHP_n^{n+1}, was proven hard for Resolution in the seminal paper of Haken [31]. In doing that, he invented an original method, *bottleneck-counting*, which is very general and has been used in many different settings so far (e.g. for reproving the separation between monotone $\mathcal{P}$ and monotone $\mathcal{NP}$.) It however took some 30 years to prove an exponential lower bound for the *weak* Pigeon-Hole Principle, where the number of pigeons can be much bigger than the number of holes [42], [43], [44].

PHP_n^m is hard for polynomial calculus, independently from the characteristic of the field [3]. $BPHP_n^m$ is trivially easy even for Nullstellensatz over a field of characteristic 0. The most interesting, and still open question, is the Nullstellensatz complexity of $BPHP_n^{n+p^k}$ over a field of characteristic p, where p is a prime number. An easy upper bound on the degree is p^k, while the best lower bound is 2^k [9].

The strongest result known up to date is that the ordinary Pigeon-Hole Principle, PHP_n^{n+1} is hard for Bounded-depth Frege. A super-polynomial lower bound was proven by Ajtai in [1]. Later on, the lower bound was improved to exponential in [5]. The result holds even we allow $Count_p^n$ axioms for some prime p [9]; the strongest result about the relation between different versions of the Pigeon-Hole Principle appear in [47].

The Pigeon-Hole Principle is trivially easy for Cutting Planes and Lovasz-Schrijver. It is also easy for the Frege systems [12]; this is probably the only non-trivial upper bound in proof complexity.

Perfect Matching and Counting mod-p Principles The **Perfect Matching Principle** is a natural generalisation of the Bijective Pigeon-Hole Principle. Given a graph $G = (V, E)$, we introduce a propositional variable x_e for every edge $e \in E$, and express the statement "G admits a perfect matching" as the following set of clauses:

$$\bigvee_{v \in N(u)} x_{\{u,v\}} \quad \text{for every } u \in V$$

$$\neg x_{\{u,v\}} \vee \neg x_{\{u,w\}} \quad \text{for every } u \in V,\ v, w \in N(u),\ v \neq w.$$

Here $N(u)$ is the set of the neighbours of u in the graph G.

This principle naturally extends to hyper-graphs. Here we shall consider only an important special case of it, **Counting mod-p Principle**, denoted by $Count_p^n$. The hyper-graph $H = (V, E)$ is the *complete p-regular hyper-graph* on the set of vertices $V = [n]$, where *n is not divisible by p*, i.e. $E = \binom{[n]}{p}$.

$$\bigvee_{a \in E,\, i \in A} x_A \quad \text{for every } i \in V$$

$$\neg x_b \vee \neg x_c \quad \text{for every } b, c \in E,\ 0 < |b \cap c| < p.$$

The strongest result known are about Bounded-depth Frege complexity of these principle can be found in [9] and [46].

Tseitin tautologies They have the distinction of being the first example of provably "hard" tautologies. Tseitin proved that these formulae, based on certain graphs, are hard for *regular resolution* as early as 1968 [51]. They are defined as follows. A graph $G = (V, E)$ and a vertex $v_0 \in V$ are given. We associate a $0-1$ variable x_e to each edge $e \in E$. The Tseitin or odd-charged tautology is the following set equations

$$\bigoplus_{v \in N(u)} x_{\{u,v\}} = \begin{cases} 0 & u \neq v_0 \\ 1 & u = v_0 \end{cases} \qquad \text{for every } u \in V.$$

Here $\oplus$stands for summation modulo 2.

Tseitin tautologies are hard for resolution [52], polynomial calculus [14] and bounded-depth Frege [28], [10]. An open problem is whether they are hard for cutting planes and/or Lovasz-Schrijver.

Random k-SAT The random k-SAT formula $\mathcal{R}_k^{m,n}$ is defined as follows. Given the set of all possible $2^k \binom{n}{k}$ k-clauses, we take a subset of size m uniformly at random. These are introduced in [17], where the following important result is proven: There exist a constant Δ_k, depending on k only, such that if $m \geq \Delta_k n$ then $\mathcal{R}_k^{m,n}$ is *unsatisfiable with high probability* (w.h.p.), i.e. exponential in n. Moreover, every resolution refutation is of exponential in n size w.h.p.

A famous open problem is the sharp-threshold conjecture: There exist a constant Δ_k, depending on k only, such that for any constant$\varepsilon > 0$ if $m \geq (\Delta_k + \varepsilon) n$ then $\mathcal{R}_k^{m,n}$ is unsatisfiable w.h.p.; if $m \leq (\Delta_k - \varepsilon) n$ then $\mathcal{R}_k^{m,n}$ is satisfiable w.h.p. The closer we have got so far is that the threshold exists for every n [27]. The sharp threshold conjecture is proven for $k = 2$ only [29]; in this case the value is $\Delta_2 = 1$. Upper and lower bounds on Δ_k are also known for any k, and especially for $k = 3$.

Only the resolution and tree-like resolution complexity of the random k-SAT is well studied [6].

Minimal Element Principle It states that every total order, defined on a finite set, has a minimal element. Its negation, written as an Second Order Existential ($SO\exists$) sentence, is the following:

$$\exists L\ \left((\forall x\, \neg L(x,x)) \wedge (\forall .x, y\ ((x = y) \vee L(x,y) \vee L(y,x))) \wedge \right.$$
$$\left. (\forall x, y, z\ (L(x,y) \wedge L(y,z)) \rightarrow L(x,z)) \wedge (\forall x \exists y\, L(y,x)) \right).$$

Here $L(x,y)$ stands for "$x < y$". The Minimal Element Principle is interesting for several reasons: It separates tree-like from general resolution; It is also a good example both of the translation from $SO\exists$ to propositional logic, described in [48], and of the general tree-like resolution complexity characterisation of the $SO\exists$ sentences, proven in the same paper. As an illustrative example, we give

the translation of the Minimal Element Principle from $SO\exists$ to propositional form:
$\forall x\, \neg L(x,y)$ is translates into $\neg l_{ii}$ for all i.
$\forall x, y\ ((x=y) \vee L(x,y) \vee L(y,x))$ into $l_{ij} \vee l_{ji}$ for all $i \neq j$.
$\forall x, y, z\ L(x,y) \wedge L(y,z) \rightarrow L(x,z)$ into$\neg l_{ij} \vee \neg l_{jk} \vee l_{ik}$ for all i, j, k.
$\forall x \exists y\, L(y,x)$ into $l_{1j} \vee l_{2j} \vee \ldots \vee l_{nj}$ for all j.

(Different versions of) the Induction Principle The (negation of) the **Induction Principle** can be encoded as

$$p_1 \wedge (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_3) \wedge \ldots \wedge (p_{n-1} \rightarrow p_n) \wedge \neg p_n.$$

Another such principle, studied in the literature, is the **House-Sitting Principle**. There are $n+1$ persons, numbered from 0 to n, and n houses, numbered from 1 to n. The i-th person owes the i-th house; the 0th person does not owe a house. Each person has to stay in some house, either his/hers own or having a bigger number. For each house, if the owner is at home, nobody else can stay there. If not, any number of persons can stay in the house. This principle can be encoded as the following set of clauses:

$$p_{01} \vee p_{02} \vee \ldots \vee p_{0n}$$

$$p_{ii} \vee p_{i,i+1} \vee \ldots \vee p_{in} \quad \text{for every } i \in [n]$$

$$\neg p_{ii} \vee \neg p_{ji} \quad \text{for every } i \in [n]\,,\ j < i$$

These principles are easy even for tree-like resolution which seems to be the weakest system on our picture. They are however hard for Nullstellensatz [37].

Partial map of Propositional Proof Systems

The known separations are summarised on the figure 1.1. There are several types of edges with the following meaning:

——▶ the source is *provably stronger* than the target, i.e. the former p-simulates later, and there is a tautology that separates them, being easy for the source but hard for the target.

- -▶ the source p-simulates the target, but neither the converse nor a separation is known.

- -▷ there is a tautology which is easy for the source, but hard for the target; neither p-simulation nor the converse separation is known.

◁—▷ the the two system are incomparable, i.e. there are tautologies separating them in both ways.

The redundant edges, i.e. that can be deduced by the transitivity, are not shown.

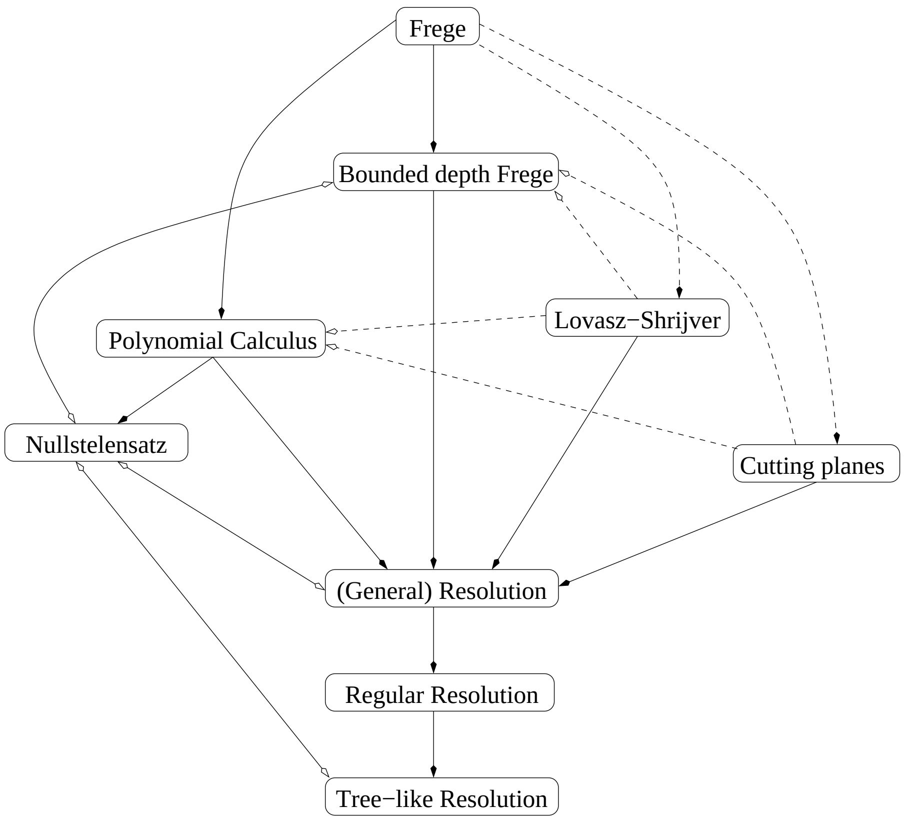


Figure 1.1: Some propositional proof systems and relation among them

1.4 Contributions of the thesis

Known results

We shall first survey the known results about the resolution complexity of the perfect matching principles.

The most important, and therefore the most studied one, is the Pigeon-Hole Principle, PHP^m_n. In his seminal paper [31], Haken showed that *any resolution proof* of PHP^{n+1}_n has to be of size $2^{\Omega(n)}$. This was the first ever truly exponential lower bound for a natural combinatorial principle and general enough proof system.

Haken's proof has been generalised and simplified in [16], [7], [11]. For quite a while, the best known result had been a $2^{\Omega\left(\frac{n^2}{m}\right)}$ lower bound from [16], thus having left the case $m = \Omega\left(\frac{n^2}{m}\right)$ as an important open problem in resolution proof complexity. An important step was done in [38], where a $2^{\Omega(n^\varepsilon)}$ lower bound on *any regular-resolution proof* of PHP^m_n was proven. Shortly afterwards, the problem was solved by Raz in [42], and further strengthen and improved by Razborov in [43], [44].

The results in [11] deserve special attention. They provide very general methodology for proving lower bounds on the *size of a resolution proof* via the clauses *width*. More precisely, the following two inequalities are proven

$$S_T(\mathcal{F}) \geq 2^{w(\mathcal{F}\vdash\oslash)-w(\mathcal{F})}$$

and

$$S(\mathcal{F}) \geq \exp\left(\Omega\left(\frac{(w(\mathcal{F}\vdash\oslash)-w(\mathcal{F}))^2}{n}\right)\right), \tag{1.5}$$

where $S(\mathcal{F})$ ($S_T(\mathcal{F})$) is the size of smallest (tree-like) resolution refutation of the unsatisfiable set of clauses $\mathcal{F}$, n is the number of propositional variables in $\mathcal{F}$, and $w(.)$ is the *width* defined as follows. The width of a clause is the number of literals in it. The width of a set of clauses F is the maximal width of a clause in the set. The width of a refutation of a set of clauses $w(\mathcal{F}\vdash\oslash)$ is the minimum taken over the widths of all possible refutations (considered as a set of clauses).

As an easy consequence, we get lower bounds for the perfect matching principles based on graphs having a good *expansion* property.

The resolution complexity of other kinds of matching principles, such as modulo-p counting principles, have not been studied so well. No resolution lower bounds were known, apart from the ones, proven for much stronger systems, ([33], chapter 12), until the very recent work of Razborov [45]. The result there is a very general lower bound for the perfect matching principle on a given hyper-graph $\mathcal{H}$ of the form

$$\exp\left(\Omega\left(\frac{\delta(\mathcal{H})}{\lambda(\mathcal{H})\,r(\mathcal{H})\log n(\mathcal{H})\,(r(\mathcal{H})+\log n(\mathcal{H}))}\right)\right), \tag{1.6}$$

where $n(\mathcal{H})$ is the number of vertices of $(\mathcal{H})$, $\delta(\mathcal{H})$ is the minimal degree of a vertex, $r(\mathcal{H})$ is the maximal size of a (hyper)edge, and $\lambda(\mathcal{H})$ is the maximal number of (hyper)edges incident to two different vertices.

As a consequences, we get two non-trivial resolution lower bounds, $\exp\left(\Omega\left(\frac{n}{(\log m)^2}\right)\right)$ for $FPHP^m_n$ and $\exp\left(\Omega\left(\frac{n}{p^2\log n(p+\log n)}\right)\right)$ for $Count^n_p$.

This thesis

There is however an important class of graphs, namely the *planar* ones, for which neither the methodology in [11], extracted in the formula (1.5), nor the result in [45], the formula (1.6) applies. Our main results, entitled

"Planar" tautologies hard for Resolution fills this gap.

Our motivation to consider the perfect matching principle on planar graphs comes from the **Mutilated Chessboard problem**. It is a well known problem in recreational mathematics: Given a chessboard with two diagonally opposite squares removed (see the left side of the figure 1.2), can it be covered by dominoes, each domino covering two neighbouring squares? The answer is obvious, "no", once one observe that the missing squares are of the same colour, therefore there are two more squares of the other colour, and a domino cover always

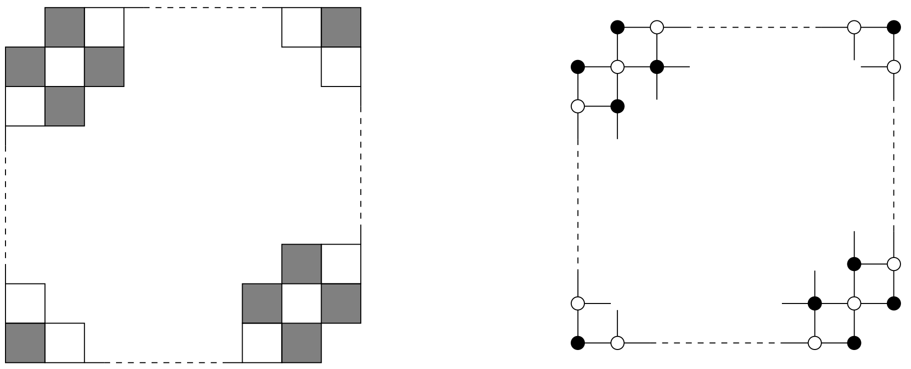


Figure 1.2: Mutilated Chessboard

consists of equal number of black and white squares, as a single piece covers two neighbours which always have different colours.

In propositional proof complexity setting of the problem, we have not ordinary but $2n \times 2n$ chessboard, and the question is: How hard is to prove the impossibility of a domino cover within a particular proof system, in our case *resolution*. The mutilated chessboard has the distinction to be the earliest problem, conjectured to be hard for automated theorem provers [36]. Clearly, it is a special case of a perfect matching problem (see the right side of the figure 1.2 for a graphical explanation). Proving a resolution lower bound for the Mutilated Chessboard problem has been a hard open problem for quite a while. As Urquhart pointed out in [53], it was "... another case where current techniques appear impotent...".

Thus our main contribution, and the main result in the thesis, is that we obtain tight resolution lower bound of $2^{\Omega(n)}$ for the mutilated chessboard problem. The main tool, we use in our proofs, is the *representation of resolution proofs as Prover-Adversary games*, introduced by Pudlak in his recent paper [40]. This gives a very "clean" proof of the lower bounds,. As an intermediate step in our proof, we introduce the concept of *tiling games*, which is of independent interest. Our reduction from a tautology to a tiling game is quite general. It allows to prove lower bounds not only for the mutilated chessboard problem, but also for a broad class of tautologies, based on *regular planar graphs.* These include not only the perfect matching principles on such graphs, but also *Tseitin tautologies.*

The second contribution of our thesis is

Tree-like Resolution complexity of the Weak Pigeon-Hole Principle Even though the tree resolution is a quite weak propositional proof system, the exact complexity of tree-resolution proofs of PHP^m_n has not been known prior to our paper. A $2^{\Omega(n)}$ lower bound was shown in [15], whereas one can construct only a $2^{O(n \log n)}$ tree proof by "unfolding" the $2^{O(n)}$ general resolution proof given in the same paper. A $2^{O(n \log n)}$ lower was proven in [32], but only for *ordinary pigeon-hole principle*, i.e. PHP^{n+1}_n.

The first and the most important contribution of our work is closing the gap. We prove a $2^{\Omega(n \log n)}$ *lower bound* on *any tree-resolution proof* of PHP^m_n, independently from m, even if it is infinity. It is *tight* up to a constant factor in the exponent or, in other words, up to *a polynomial transformation.* In order to prove the result, we introduce a new method for proving lower bounds on tree-resolution proofs. It is more general than the existing one (see e.g. [48]). The latter works only for balanced proofs, whereas any tree-like resolution proof of PHP_n^m is highly *unbalanced* as shown in our paper.

The second result in this paper is considering *the worst tree regular resolution* proofs of PHP_n^m. This is for the first time, *the worst case* proof complexity has been considered. We prove an *upper bound* of $2^{O(n \log m)}$, which is non-trivial, as there are mn variables, and one can therefore expect the worst case to be as bad as 2^{mn} (we consider of course only proofs which do not contain vacuous weakening of axioms). This has the following very interesting consequence: Consider $PHP_n^{poly(n)}$, i.e. m is some polynomial in n. The optimal and the worst regular tree-resolution proofs of $PHP_n^{poly(n)}$ are *polynomially related*, and so are any two *random* regular tree-resolution proofs. This also has an implication in automated theorem proving, as it shows that there are natural problems for which any *DLL-based* proof search heuristic is as good as any other. The same applies to the pigeon-hole principle, encoded as a $SO\exists$ sentence. It is easy to see than any such encoding would result in $PHP_{q(n)}^{p(n)}$, where $p(.)$ and $q(.)$ are polynomials, and n is the size of the (finite) universe. Therefore, there exists a non-empty class of $SO\exists$ tautologies, such that any two tree resolution proofs of a sentence from the class are polynomially related.

The last result of the thesis is

Resolution width-size trade-offs for the Pigeon-Hole Principle We first prove an upper bound, i.e. construct a resolution proof of $WPHP_n^m$ of size $2^{O\left(\frac{n \log n}{\log m} + \log m\right)}$ for any m and n. Such an upper bounds have been known so far only for the two extreme cases $m = n + 1$ and $m = 2^{\sqrt{n \log n}}$ (see [15]). Ours matches these, and, moreover, we believe that it is the exact resolution proof complexity of $WPHP_n^m$ for all m and n (recall that the best lower bound known so far is $2^{\Omega\left(\frac{n}{(\log m)^2}\right)}$ from [45]).

We also prove a $2^{\Omega(n)}$ lower bound on any resolution proof of the weak pigeon-hole principle, $WPHP_n^m$, when the width is bounded by $\left(\frac{1}{16} - \varepsilon\right) n^2$. Unlike the general lower bound, it holds independently from the number of the pigeons, m.

These two results not only give a resolution width-size trade-off for the Weak Pigeon-Hole Principle, but also have the following interesting consequence. Letting $m = 2^{\sqrt{n \log n}}$, we can use it to construct a tautology on N variables which is provable in resolution within polylog (N) width. Any such proof however is of super-polynomial in N size. At the same time, there is a proof of poly (N) size and width N. This is asymptotically, i.e. modulo the degrees of the polynomials, the best resolution width-size trade-off, one could hope to prove.

Chapter 2

“Planar” tautologies hard for Resolution

We prove exponential lower bounds on the resolution proofs of some tautologies, based on rectangular grid graphs. More specifically, we show a $2^{\Omega(n)}$ lower bound for any *resolution proof* of *the mutilated chessboard problem* on a $2n \times 2n$ chessboard as well as for *the Tseitin tautology* based on the $n \times n$ *rectangular grid graph*. The former result answers a 35 year old conjecture by McCarthy.

2.1 Introduction

In the paper, we prove an exponential lower bound for any resolution proof of the mutilated chessboard problem as well as for the Tseitin tautologies on a rectangular grid graph.

Exponential lower bounds for resolution are known for matching problems based on the complete bipartite graph $K_{n+1,n}$ as well as for a special class of graphs, namely expanders (see [31], [53], [16]). Exponential lower bounds for Tseitin tautologies are also known for expander graphs only [51]. In the recent paper [11], a common framework is given that generalises and simplifies all the known proofs. Unfortunately, it does not work for tautologies based on planar graphs.

Thus our main contribution is that we obtain exponential lower bounds for tautologies, based on grid graphs. The main tool, we use in our proofs, is the representation of resolution proofs as Prover-Adversary games. It is introduced by Pudlak in his recent paper [40]. On a technical level, our contribution is a new way to introduce randomness in Adversary’s strategy (although Pudlak, himself, speaks about “super-strategy” rather than “randomised strategy”). In doing so, we introduce the concept of tiling games. It turns out that the combination of our reduction of the original problems to tiling games and Pudlak’s idea of considering proofs as games gives very “clean” proofs of the lower bounds.

The paper is organised as follows. First, we define the two problems and explain briefly Pudlak’s idea of considering resolution proofs as games. We then introduce tiling games and prove lower bounds for them. Finally, we show the reduction from the original problems to tiling games.

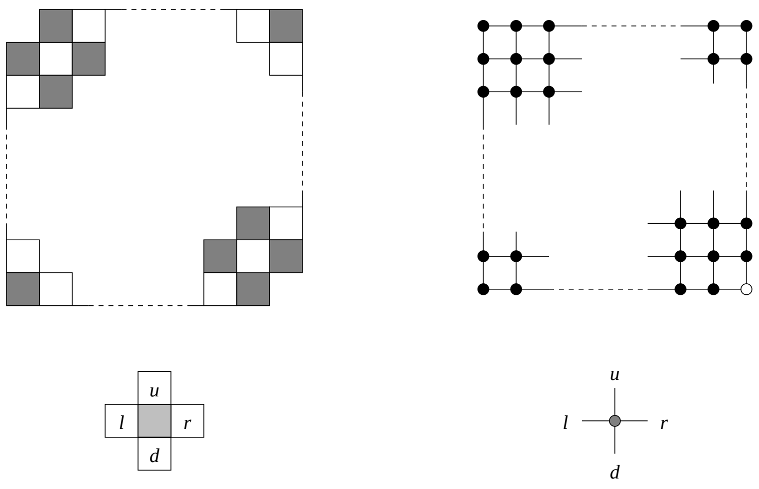


Figure 2.1: The two original problems

Mutilated chessboard This problem has the distinction to be the earliest proposed hard problem for theorem provers [36]. The problem is the following: given a $2n \times 2n$ chessboard with two diagonally opposite squares missing (see the left side of Fig. 2.1), prove that it cannot be covered with dominoes. We can consider it as a matching problem (the left part of Fig. 2.2): squares are vertices of the graph, and there is an edge between every two neighbouring squares. Thus one component of the bipartite graph consists of black squares and the other consists of white ones. Two missing squares are of the same colour which implies one of the components in the graph has two more vertices than the other. That is why there is no perfect matching, i.e., dominoes tiling of the mutilated chessboard.

The formalisation of the problem as a set of clauses is as follows. For every square, we introduce (at most) four variables u, r, d, l corresponding to the four possible ways of covering a square by a domino. We then write down the following clauses, saying that every square is covered exactly once:

1. $\{u, r, d, l\}$
2. $\{\overline{u}, \overline{r}\}$, $\{\overline{u}, \overline{d}\}$, $\{\overline{u}, \overline{l}\}$, $\{\overline{r}, \overline{d}\}$, $\{\overline{r}, \overline{l}\}$, $\{\overline{d}, \overline{l}\}$

Whenever a variable does not make sense, i.e., a domino, going outside the chessboard, we replace the corresponding variable by "false".

Tseitin tautologies on grid graphs The definition of the problem is as follows. Given a undirected graph, we attach a propositional variable to every edge. We also select one vertex and label it by "true", all others are labelled by "false". We require the exclusive-or of all the adjacent edges of every vertex to be equal to its label. Obviously, this is impossible as every variable occurs exactly twice in the exclusive-or part of these equations, but the exclusive-or of all the labels is "true".

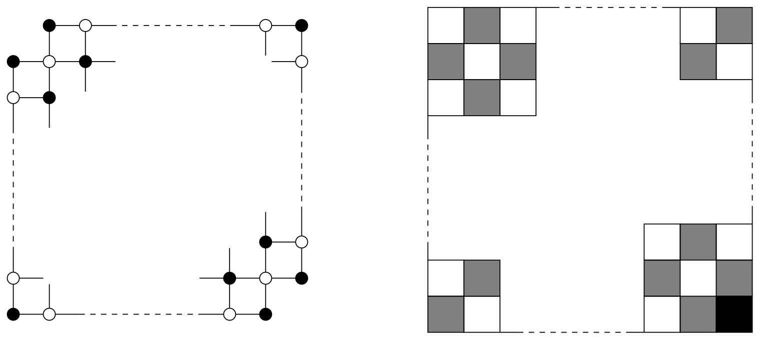


Figure 2.2: The “reverse” formulations

On a $n \times n$ rectangular grid graph, we colour white one of the corners, and all the other vertices are black (see the right side of Fig. 2.1). We then write the following set of clauses:

$$u \oplus r \oplus d \oplus l = \begin{cases} false & \text{for all the black vertices} \\ true & \text{for the only white vertex} \end{cases}$$

- $\{\overline{u}, r, d, l\}$, $\{u, \overline{r}, d, l\}$, $\{u, r, \overline{d}, l\}$, $\{u, r, d, \overline{l}\}$, $\{u, \overline{r}, \overline{d}, \overline{l}\}$, $\{\overline{u}, r, \overline{d}, \overline{l}\}$, $\{\overline{u}, \overline{r}, d, \overline{l}\}$, $\{\overline{u}, \overline{r}, \overline{d}, l\}$ for a black vertex
- $\{u, r, d, l\}$, $\{\overline{u}, \overline{r}, d, l\}$, $\{\overline{u}, r, \overline{d}, l\}$, $\{\overline{u}, r, d, \overline{l}\}$, $\{u, \overline{r}, \overline{d}, l\}$, $\{u, \overline{r}, d, \overline{l}\}$, $\{u, r, \overline{d}, \overline{l}\}$, $\{\overline{u}, \overline{r}, \overline{d}, \overline{l}\}$ for the white vertex

Again, all the variables that do not make sense are replaced by “false”.

There is another, chessboard-style formulation of the problem. Given a chessboard, tile it by dominoes, such that every square is covered by even number of tiles and one of the corners is covered by odd number of tiles. The formulation is illustrated on the right side of Fig. 2.2.

2.2 Preliminaries

Resolution We first give some definitions. A *literal* is either a propositional variable or the negation of propositional variable. A *clause* is a set of literals. It is satisfied by a truth assignment if at least one of its literals is true under this assignment. A set of clauses is *satisfiable* if there exists a truth assignment satisfying all the clauses.

Resolution is a proof system designed to *refute* given set of clauses, i.e., to prove that it is unsatisfiable. This is done by means of the resolution rule

$$\frac{C_1 \bigcup \{v\} \quad C_1 \bigcup \{\neg v\}}{C_1 \bigcup C_2},$$

i.e., we can derive a new clause from two clauses that contain a variable and its negation respectively. The goal is to derive the empty clause from the initial ones. For technical reasons only, we use the weakening rule

$$\frac{C}{C \bigcup \{v\}},$$

even though its use is not essential and can be avoided.

Anywhere we say we *prove* some proposition, we mean that first we take its negation in a clausal form and then use resolution to refute these clauses.

There is an obvious way to represent every resolution refutation as a directed acyclic graph whose nodes are labelled by clauses. The sources, i.e., the vertices with no incoming edges, are the initial clauses, and the only sink, i.e., the vertex with no outgoing edges, is the empty clause. If we reverse the directions of the edges, and consider the sink as a root and the sources as leaves we get a *branching program*. It is easy to see that it solves the following *search problem*, associated with the given set of unsatisfiable clauses: given an assignment, find a clause that falsifies it. Unfortunately, the reverse is not true, that is we cannot convert any branching program, solving the search problem, into a resolution proof.

As a matter of fact, there are polynomial-size branching programs, solving both problems from the paper. Of course, this does not contradict to our main result, as it shows that these branching programs cannot be transformed into resolution proofs.

In our proof we essentially use a representation of resolution proofs as *Prover-Adversary game*s, called further *Resolution Games*. This approach is introduced by Pudlak in [40]. A brief description follows.

Proofs as Games There are two players, named *Prover* and *Adversary*. An unsatisfiable set of clauses is given. Adversary claims wrongly that there is a satisfying assignment. *Prover*'s task is to convict him in lying. A *position* in the game is a partial assignment of the propositional variables. The game start from the empty position. Prover has two kind of moves:

1. She queries a variable, whose value is unknown in the current position. Adversary answers, and the position then is extended with the answer.

2. She forgets a value of a variable, which is known. The current position is then reduced, i.e., the variable value becomes unknown.

The game is over, when the current partial assignment falsifies one of the clauses. Prover then wins, having shown a contradiction.

As she can always win, simply querying consecutively all the variables and not forgetting anything, Adversary's task is to force Prover to use *big memory*, "big" meaning exponential in the number of variables. We assume that she keeps her strategy as a *list of ordered pairs* (position, move), where "position" and "move" have their natural meaning. Thus, it is enough for Adversary to use a strategy, which ensures big number of different possible positions, no matter how Prover plays.

The reduction from a resolution proof to Resolution Game should now be clear. Although trivial, we will not explain it here and refer to [40] for all the details. We should however note that a *deterministic* Adversary's strategy corresponds to a *single path* in the proof's graph. Therefore, he has to use a *randomised strategy* (called "super-strategy" in Pudlak's paper) in order to enforce a *big enough subgraph.*

It is very important to make the following conventions: Every time we say "Prover's strategy", we mean *wining strategy*, as only a wining strategy corresponds to a resolution proof. Every time we say "Prover ... in order to win" we also mean *wining strategy*, i.e., Prover is not interested in wining a single game, but any game, no matter how Adversary plays.

We can finally state the main results and explain informally the main ideas behind the proofs.

Main results and outline of the proofs. We prove the following two theorems:

Theorem 2.1 *Any resolution proof of the Mutilated Chessboard problem is of size* $2^{\Omega(n)}$.

A weaker version of the above theorem, with $\sqrt{n}$ in the exponent, is proven independently in [2].

Theorem 2.2 *Any resolution proof of Tseitin tautologies, based on* $n \times n$ *rectangular grid graph, is of size* $2^{\Omega(n)}$.

The general idea of the proofs is following:

We consider Resolution game. Clearly, Prover's queries are pairs of neighbouring squares, and Adversary's answers are dominoes, covering these pairs. A domino can be either "yes" or "no", with the natural meaning. We divide the chessboard into non-overlapping *constant-size* squares called *zones.* During the game every zone is either completely empty or completely covered by dominoes by Adversary. Here "completely" means the entire zone, except possibly few squares on the borders. In the first, *randomised*, phase of his strategy, Adversary first constructs many covers of the zone, depending on all the possible shapes of its neighbouring zones, and he then picks one of them *at random* and remembers it. These covers satisfy certain conditions that will be explained later in the paper, when proving the results. The second, *deterministic*, phase is the real game. When Prover queries a variable, i.e., a domino, inside an empty zone, Adversary puts the cover, already chosen in the first phase. He does not however reveal the cover to Prover, but only answer the question consistently with the cover. If Prover forgets *all the queried* variables inside a covered zone, Adversary removes the cover, so that the zone becomes empty again. Thus a zone is nonempty if and only if it contains at least one "significant" variable (the exact meaning of this is given in the detailed proof), whose value is kept by Prover. There are two main points in our proof:

1. Prover has to remember $\Omega(n)$ variable values at some point in the game in order to win. That is in any resolution proof we have a clause, containing linear in n number of variables. This can be proven on somewhat higher

level, depending on the connection between zones, but not on their specific covers or particular shapes. A nice abstraction of that is *Tiling Games*. They are considered in a separate section.

2. Every two values, kept by Prover and belonging to different zones are *independent* of each other. Moreover, given any value, kept by Prover, there is a *constant probability*, bounded away from 0 and 1, that the value *agrees* with the first, randomised phase. These properties depend on randomised phase only, and on some specific properties of the zone covers, designed there. This can be thought as a *reduction* of Tilling Game to Resolution Game.

It is not hard to see that these two conditions imply an exponential lower bound on Provers's memory, and therefore on any resolution proof of the corresponding problem.

The rest of the paper is organised as follows. We first introduce *Tiling Games*. They allow us to work on the level of zones only, when proving the first main claim. We also prove an exponential lower bound for these games. After that, we show a reduction between Resolution Games and Tiling Games that preserves the lower bound. This proves the second main point.

2.3 Tiling games

In this section, we introduce *tiling games* and prove some results about their complexity.

Definition of a general tiling game. The *board* of the game consists of $m \times m$ squares. Any of them is a perfect square, except the bottommost right one that has a dent on its right side. The board is shown on the left of Figure 2.3. The *tiles* of the game are squares. Their sides are of three kinds: "flat

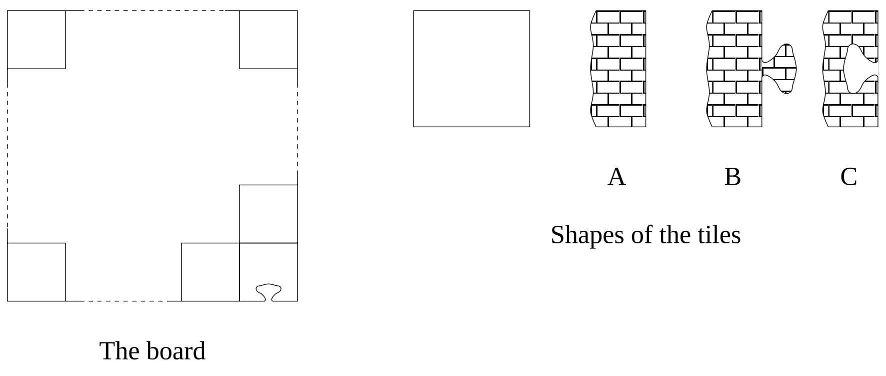


Figure 2.3: The tiling games

wall", "hump", and "dent", as shown on the right of Figure 2.3, pictures A, B, and C, respectively. Apart from its *shape*, every tile has also a *colour*, either red or blue. When we say *a general tiling game*, we mean a game where any set of shapes is allowed, whereas to get *a particular tiling game*, we fix this set. Thus, a (particular) tiling game is completely determined by its set of tiles. In both cases, every colour, either red or blue, is allowed for every shape.

In what follows, we will however consider only sets of tiles, having the property that they *cannot completely cover* the board. In particular, we put the restriction that the difference between the number of dents and the number of humps has to be even for any tile from the set. A trivial parity argument then implies the impossibility of tiling the board. We can also note that there are 41 such tiles, and therefore 2^{41} possible tiling games, as any subset of tiles defines a different one. We will however be interested in only two of them.

There are two players, whose names are *Prover* and *Adversary.* Adversary claims that there is a tiling of the board. Prover's task therefore is to force a *clear* contradiction, i.e., a tile on the board, which is *inconsistent* with one of its neighbours. In that case, she wins the game, which is played as follows: At the beginning the table is empty. At any round Prover starts by doing one of the following two:

1. She asks Adversary to put a tile on a particular square. He does so, and the round is over. We assume that Prover has infinite number of tiles of any kind (that is any allowed shape and any colour).
2. She removes any tile, already on the board. Adversary does not do anything, and the round is over.

The game is over, when Adversary is not able to play in the first case. That is, there is no tile, whose shape is consistent with the tiles, already on the board (note that the colour does not play any role here). Prover can always win by simply asking about all the squares and not removing anything. Adversary, knowing this, does not hope to play forever. His task instead is to force Prover to use *big memory*, no matter what she does.

Therefore, we need finally to explain how Prover "memorises" her strategy: The strategy is kept again as a *list of ordered pairs* (position, move), where "position" and "move" have their natural meaning. It is now clear how Prover plays: In the beginning, she finds a pair, having its position-part empty. She then makes the move-part of the pair. A new position appears. Prover finds a pair, having the new position in its position part, and then makes the move-part, and so on... We need also the restriction, that every two pairs from Prover's list have to have different position-parts, that is Prover's strategy is *deterministic.*

We can now explain how Adversary enforces the use of *big memory.*

Adversary's strategy and general lower bounds. First of all, let us note, that Adversary's strategy *cannot be deterministic*, as Prover can query about all the squares in some fixed order, never removing anything from the board, thus wining the game in m^2 memory.

We will now describe a *randomised* strategy, which is *optimal* against any Prover's strategy.

The first, *randomised*, part is very simple. It involves *the colours only.* We choose the colour for *all the squares* independently, at random, with equal probabilities of 1/2. During the game, when asked to put a tile on a particular square, Adversary always uses the initially chosen colour.

The second part is completely *deterministic.* It involves *the shapes of the tiles only.* To explain it, we need some definitions.

Definition 2.3 *Given a position, a* ***bad square*** *(for this position) is a square, such that the board, except this square, can be tiled. The* ***bad region*** *is the set of all the bad squares.*

In general, it is not even clear that a bad square exists for any position. From now on, we shall however consider only tiling games, which satisfy the following

Property 2.4 *The bad region for the starting position, i.e., an empty board, is* the entire board*, itself.*

Informally speaking, we would like to be able to move the "problematic" square from the south-eastern corner to any other position. We can make the following trivial observation:

Proposition 2.5 *Given a position where the bad region consists of two or more squares, Prover cannot win immediately, i.e., at this round.*

The next observation, although simple, is essentially the second part of Adversary's strategy.

Proposition 2.6 *Adversary can play in such a way, that the size of the bad component decreases by at most a constant factor after every round.*

Proof Let us denote the bad region by B. If Prover removes a tile from the board, the size of B remains the same or increases. Let us suppose now that Prover asks Adversary to put a tile on the empty square s. Adversary then tries all possible tiles $t_1, t_2 \dots t_k$, i.e., shapes consistent with the non-empty neighbours of s, as the colour has already been decided in the first part of the strategy. For each of these k possibilities, we denote the new bad region by $B_1, B_2 \dots B_k$. Let us now observe that any bad square for the initial position, b, has either to be s (if it is bad itself) or to belong to some B_j. The latter holds, because in tiling the entire board except b, there is a tile among the t_js, put on s, and then b certainly is in the corresponding B_j. Therefore, we have

$$|B_1| + |B_2| + \dots + |B_k| \geq |B| - 1.$$

It is now clear that Adversary has to take the most natural decision, that is to maximise the size of the new bad region. In this case

$$|B_{new}| \geq \frac{|B| - 1}{k} \geq \frac{|B|}{2k}.$$

This completes the proof, as k is less or equal to the number of all possible shapes of tiles, which is a constant (at most 41, as already mentioned). □

An important consequence is the following fact.

Proposition 2.7 *In any play of a tiling game there has to be a point, when* the bad region area *is* αm^2 *for some constant* α*,* strictly *between* 0 *and* 1*. At the same round* the border *of the bad area has to be of* length βm *for some positive constant* β *depending on* α*.*

Informally speaking, there must be a point, when the bad region is "big", i.e., quadratic in m. Naturally, in order to "surround" such a big area, we need a "big", i.e., linear in m, border. Of course, we need first to rigorously define the concepts mentioned in the statement, even though their meaning is intuitively clear.

Two squares on the board are *neighbours* if they have a common side. A *region* is an arbitrary set of squares. The *border* and the *complement* of the region R, $\partial(R)$ and $co(R)$, are defined as follows: $\partial(R)$ is the set of squares, having the property that each element in-there has a neighbour in R. $co(R)$ is all the rest, i.e., it contains every square that is in neither R nor $\partial(R)$. The *closure* of R is $\overline{R} = R \cup \partial(R)$. When we say "area" and "length", we really mean "number of squares".

We can now prove the proposition, itself.

Proof Let us observe that the area of the bad region goes from its initial value m^2 to 0 at the end, as it is a wining play for Prover (Proposition 2.5). Consider the first round, after which the area drops below $m^2/2$. After that round, it has to be bigger than $m^2/(2 \times 2 \times 41)$, according to Proposition 2.6. This proves the first part, with $\alpha \in [1/164, 1/2]$.

For the second part, we will use the following lemma, whose proof is given in the appendix.

Lemma 2.8 *For any region* R*,* $|\partial(R)|^2 \geq \min\left\{\left|\overline{R}\right|, \left|\overline{co(R)}\right|\right\}$.

Clearly, it implies the second claim in the proposition, with $\beta = \min\left\{\sqrt{\alpha}, \sqrt{1-\alpha}\right\}$. In our special case $\beta = \sqrt{\alpha}$, as $\alpha \leq 1/2$. □

Let us summarise what has been done so far: We have considered **a general tiling game**, under the only (rather weak) assumption that the bad region is the entire board at the starting position. We have proven that in any game played, there is a point, when the bad region has to have **a border linear in** m . We can note that we have not used the colours of the tiles in any way.

We can now formulate our second assumption.

Property 2.9 *In any position in the game, the number of tiles on the board is* linear *in the length of the border of the bad component.*

Adding this general, though still weak, assumption, to the first one, we can easily prove *an exponential* in m lower bound on Prover's memory.

Theorem 2.10 *In a tiling game, satisfying properties 2.4 and 2.9, any Prover's wining strategy is of size* $2^{\Omega(m)}$*.*

Proof According to Lemma 2.7, there is a point in the game, when the bad area is of size quadratic in m. At the same point, the border has to be of size$\Omega(m)$. By Property 2.9, the number of tiles on the board is $\Omega(m)$, too. The probability, that this position is consistent with the first part of Adversary's strategy, random colouring, is $1/2^{\Omega(m)}$. Therefore Prover has to have at least $2^{\Omega(m)}$ different position in the memory, as otherwise, there would be a choice of the colours, for which she does not win.□

At the end, let us note that our two lower bounds on the size of both a position in the game (linear) and Prover's memory (exponential) are tight. A simple divide-and-conquer algorithm yields *upper bounds* of $O(m)$ for the size of the position at any time and $2^{O(m)}$ for the memory Prover needs.

What remains to be done is to prove that our two assumptions are indeed correct for the concrete tiling games we are interested in.

Length lower bounds for particular games We shall first define the two games.

1. **Tseitin** is the tiling game, having as a set of tile-shapes all the shapes for which the *difference* between the number of dents and the number of humps is *even.*

2. **Mutilated Chessboard** is the tiling game, having as a set of tile-shapes all the shapes for which the number of dents *equals* the number of humps.

Clearly, Mutilated Chessboard game is a restricted version of Tseitin game. Thus, every lower bound for the former game applies to the latter, too. On the other hand, one could expect that proving lower bounds for Mutilated Chessboard game would be much harder. It is indeed the case, as the reader will see. This is the reason, we spend most of the rest of the paper on Mutilated Chessboard game rather than on Tseitin one.

First of all, let us observe that both games trivially fulfill the first assumption, saying that initially, the bad region is the entire board.

Thus only the second assumption, namely that at any round, the number of tiles on the board is linear in the border-length of the bad region, is to be checked.

We start with the easier, Tseitin, tiling game. In this case, the deterministic part of Adversary's strategy can be simplified. The key observation is that we can always keep the bad region *isolated.*

Definition 2.11 *The bad region is* ***isolated*** *iff it is* separated *by tiles from any other region of the board, consisting of empty squares. In other words, a neighbour of a bad square is either another bad square or a tile, but never a square, which is not bad.*

In general, we need to consider all the connected components of empty squares. We can keep the following invariant: exactly one of them is bad and the others are *good*, i.e., they can be tiled. It can be easily proven that the only component having one more dent that humps is the bad one, and moreover any component, having equal number of dents and humps, is good. When Prover asks a question, she may disconnect the component, where the question is, into (at most four) other components. Conversely, if Prover removes a tile from the board, she may join some previously disconnected components into a new one. Adversary needs to be careful only in the case when the Prover's question disconnects the current bad component. If so, Adversary answers in such a way that *the biggest* of the new obtained components becomes *bad* and the rest become good. It is easy to see that Adversary can always do that by a tile, consistent with the neighbours of the queried square. After any round of the game, the bad component can decrease by a factor of four at most, thus Proposition 2.6 holds, and so does Proposition 2.7 with $\alpha \in [1/16, 1/2]$ and $\beta = 1/4$. As *the bad region* is always *isolated*, its border consists of tiled squares only. Therefore the second assumption is fulfilled, and Theorem 2.10 then applies, giving us the following:

Lemma 2.12 *Prover needs at least* $2^{\frac{m}{4}}$ *memory cells in order to win Tseitin tiling game.*

Let us now consider the other tiling game, Mutilated chessboard one. It is now not so easy as before, as *the bad region* does *not* need to be *isolated*. This

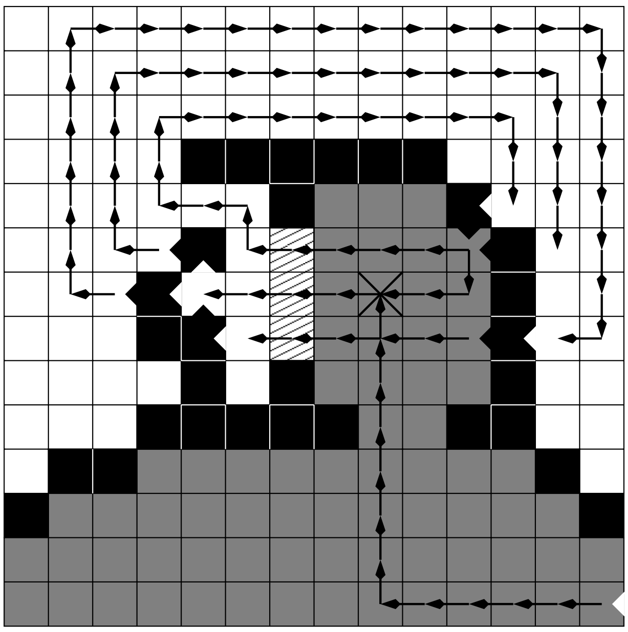


Figure 2.4: Tiling to flow correspondence

is illustrated on Figure 2.4. Black squares are the tiled ones. We call them also *marked* squares. The bad region is shown in gray. Its border contains not only marked squares, but also some empty squares, which are shown dashed. One could, in general, think that it is possible, in some clever way, to "surround" a "big" bad area, using only "few" tiles. Our intuition however tells us, that it should be impossible. If the border of the bad region is coarse, it would be possible to "push" a problematic square through it, thus "extending" the bad region, which is impossible by its definition. The following argument formalises the intuition.

We shall first explain the connection between of a position in the Mutilated Chessboard game to max-flow in a graph. The vertices of the graph are the empty squares and there is an edge between any two neighbours. We call *a source* an empty square such that there are more dents inside it than the dents in the neighbours, adjacent to the considered square, i.e., the humps of the square if it were tiled. The difference between the number of these two numbers is the *capacity* of the source. On the picture, there are four sources of capacity one and one source of capacity two. They are the squares with outgoing arrows only. In general, when counting them, we take into account the capacity, so that we can say that there are six sources on the picture. If we exchange "dents" by "humps" and vice versa in the above definition, we get the definition of a *sink*. There are five sinks on the picture that are exactly the squares with incoming edges only. Obviously, the number of sources is always greater by exactly one than the number of sinks during the game. It is clear that the bad squares and only they have the following property: if we choose one of them as a sink, so that the number of sources equals the number of sinks, there is a max-flow of capacity equal to the number of sources. An example is given on Figure 2.4, where the crossed square is chosen and a max-flow (of value 6) is shown by the arrows. It is also straightforward to convert the flow into the corresponding tiling an vice versa.

Proposition 2.13 *In any position of Mutilated Chessboard tiling game, the number of empty border squares is a constant fraction of the total number of border squares.*

Proof We now consider the border of the bad region. What we need to prove first is that the empty squares, belonging to it, are not "too many", namely they are less than the number of sources.

The proof uses a max flow - min cut argument. Let us introduce a new, artificial vertex A, and a directed edge from every empty border square to A. Let us put the new vertex as a sink of capacity one, and denote the number of sources by k. We now claim that there is no flow of value k in the new graph. Suppose there were. Then it had to go trough one of the new edges. But then we could "stop" it in the corresponding border squares that would imply this square is bad - a contradiction.

Since the max flow equals the min cut, there has to be a cut of size less than k. Let us take one such cut, and call the *sources side "right"* and the *sinks side "left"*.

We first need to show that the artificial vertex A is not contained in the cut[1] (note that a cut can, in general, contain not only edges but also vertices, having capacities). Suppose that the cut contained A. Consider the part of the cut when restricted to the original graph, i.e., before adding A and the edges

[1]We thank Mikhail Alekhnovitch for pointing out that we have forgotten to include this part of the argument in an earlier version of the paper.

from any empty border square to A. This part of the cut must be of capacity at least $k-1$ as it separates the k sources from $k-1$ sinks, and there is a max-flow of value $k-1$ in the original graph. Therefore any cut containing A is of capacity at least k. This implies that it cannot be minimal as there is no flow of value k in the new, containing A graph.

We can now see that all the bad squares are on the right side. Suppose there were at least one on the other side. But this implies that the size of the cut is greater or equal to k, because there is a flow of size k if a bad square is taken as a sink - a contradiction.

Let us denote the sets of border squares on the left/right side by L/R respectively. What we have proven so far is shown on Fig. 2.5. We should however

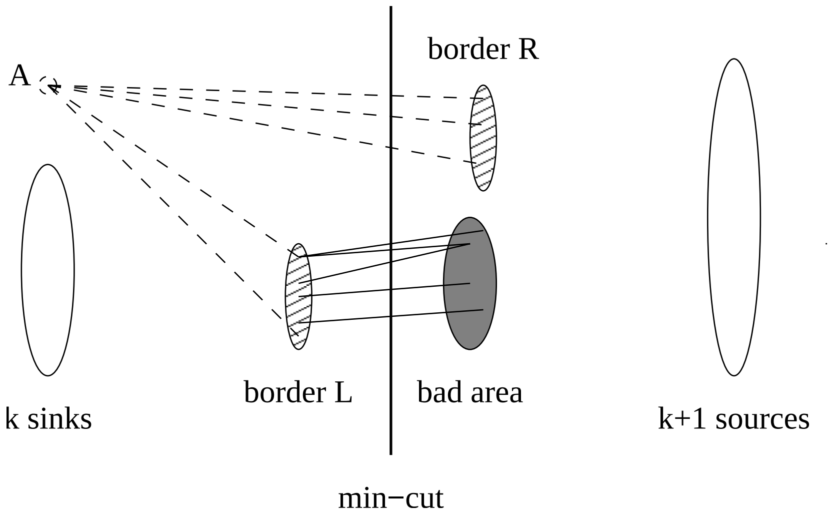


Figure 2.5: Max flow-min cut argument

note that there are two properties, which we “ignore” in our proof, because we do not need them. First, as a matter of fact, it can be proven that all the border squares are on the left side of the cut. Therefore, $R = \emptyset$, and the cut contains no artificial edges. Second, it can be shown that the bad area is connected.

The cut we consider contains at least the following edges: exactly one edge from every vertex in R to the new vertex and at least one edge from every vertex in L to some bad vertex (as every vertex in L is on the border of the bad area). This implies that $|L|+|R| < k$, that is the number of border squares is less than the number of sources.

We can finally prove that the second property, saying that at any round, the number of tiles on the board is linear in the border-length of the bad region, is fulfilled. Given a position on the board, denote the border-length of the bad region by l. Suppose the *number of sources* is *at most* $l/2$. There are then at most that many empty squares on the border, thus the number of marked squares, that is the number of tiles on the board, is at least $l/2$. Suppose now the opposite, the *number of sources* is at *least* $l/2+1$. Observe now that every marked square generates at most two sources, as any tile has at most two humps. Thus, there have to be at least $l/4$ tiles on the board. This completes the argument.□

As proven before, the above proposition implies

Lemma 2.14 *Prover needs* $2^{\Omega(m)}$ *memory cells in order to win Mutilated Chessboard tiling game.*

2.4 Reduction

In this section, we show how to reduce *Resolution game,* played on a c*hessboard,* into *Tiling game.* To understand what "reduction" really means in this context, we need to look at Fig. 2.6.

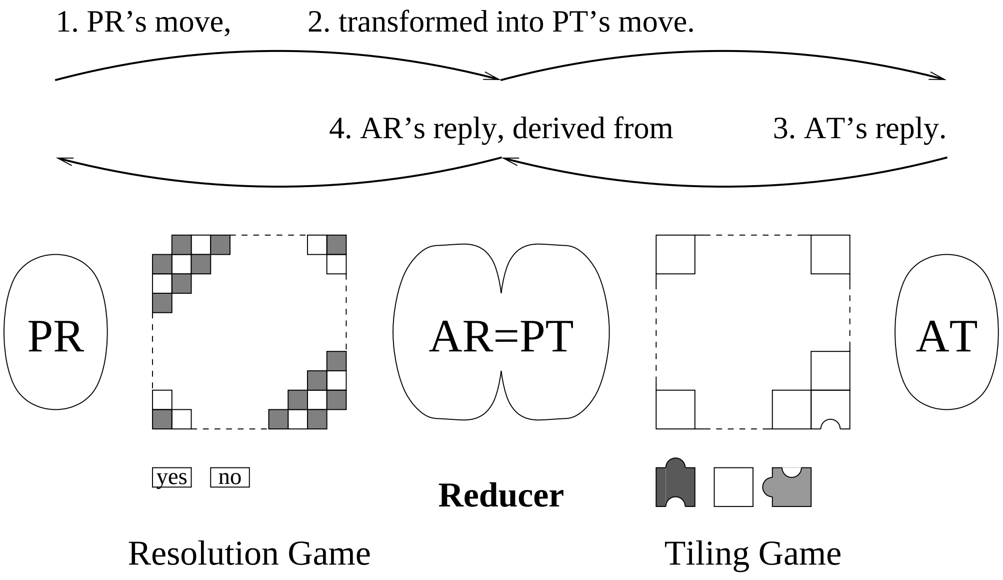


Figure 2.6: The general schema of the reduction

The resolution game is played by Prover Resolution (PR) and Adversary Resolution (AR), while the tiling game is played by Prover Tiling (PT) and Adversary Tiling (AT). We think of AR and PT as a single person, named *Reducer*, who carries the *reduction.* As shown on the figure, he first looks at the PR's move in the resolution game. He then transforms it into PT's move in the resolution game and plays it there. After having got AT's reply, Reducer transforms it into AR's move and replies by it to the initial PR's move in the resolution game.

Thus, one can think that the real game is played between PR and AT. We already have a particular AT's strategy which forces an exponential lower bound on any PT's strategy. We will prove, that this important property can be carried trough the reduction, that is to imply an exponential lower bound on any PR's strategy, too. We will only consider the reduction of Mutilated Chessboard problem, which is technically harder. That is why, we describe it in full detail, leaving the reduction for Tseitin tautologies to the reader.

Mutilated chessboard As we mentioned before, we first divide the chessboard into non-overlapping *constant-size squares*, called further *zones*. In our rigorous proof, we use 48×48 squares. A *zone* in Mutilated Chessboard problem corresponds to a *square* in Mutilated Chessboard tiling game. We also "move" one of the missing squares near to the other as shown on Fig. 2.7 for a $(48n + 2) \times (48n + 2)$ chessboard. We first define what a *zone* is. It is a "big",

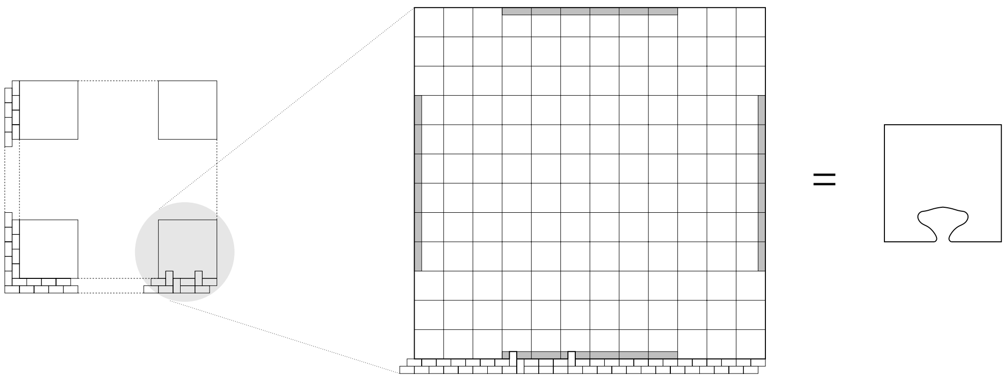


Figure 2.7: The reduction: zones, corresponding to tiles

48×48, square, with "few" small squares cut off. We need to explain how the missing squares can exactly appear.

We divide a zone into 12×12 smaller, 4×4, squares, further called *sub-zones*, as done in the middle of Figure 2.7. *Missing squares* can only appear on the *sides* of a zone, inside the four dashed "bands" which are the border part of the four gray five-sub-zone areas. Moreover, there are only the following possible shapes:

1. No missing squares (Fig. 2.8A). This corresponds to a *flat wall*, in the tiling game.

2. Two neighbours, of *different colour*, belonging to a sub-zone, and *not* being the two *middle* squares (Fig. 2.8B). In the tiling game, this corresponds to a *flat wall*, similar to the previous case.

3. Two squares of *the same colour,* belonging to sub-zones, that are a sub-zone away from each other (Fig. 2.8C). *Two black* missing squares correspond to a *hump* in the tiling game*,* whereas *two white* missing squares correspond to a *dent.*

4. Four missing squares, that are a *combination* of the previous two cases, and, moreover, no two mutilated sub-zones can be neighbours (Fig. 2.8D). Again, if *two more white* squares than black ones are missing, this is *a dent* in the tiling game. The symmetric case, i.e., *two more blacks*, corresponds to *a hump.*

Figure 2.8 shows all possible shapes of a *zone border.* They imply all possible *connections* between two *neighbouring zones.* In particular, two neighbours can

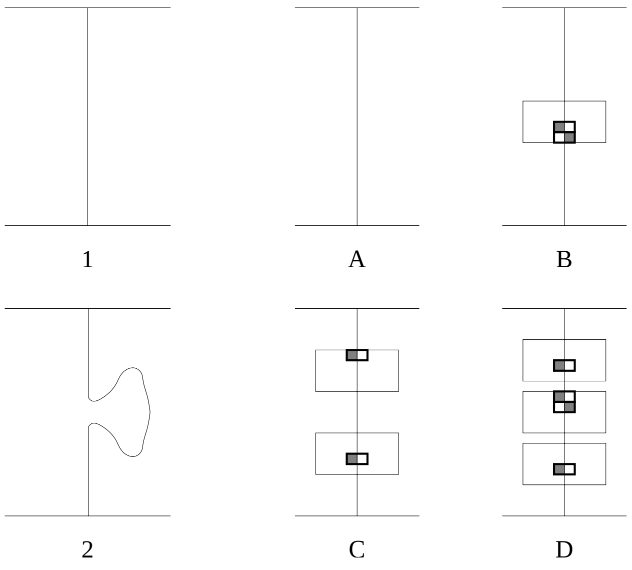


Figure 2.8: The reduction: possible shapes of zones and connections between them

have only 0, 2 or 4 dominoes in common, and, moreover, these can only appear as explained above.

We are now almost ready to explain the essence of the section, namely Reducer’s algorithm. The last, but the most important concepts, we need, are the different kind of questions we have in Resolution Game. At any round in the game, we have a partial tiling of the chessboard. The tiling satisfies the condition that any zone is either *completely empty* or *completely covered* by dominoes from Adversary’s point of view. There are the following three kind of Prover’s questions:

Definition 2.15 ***Dummy*** *question is a question about a domino, connecting two neighbouring zones, and within* 3 *sub-zones (that is* 12 *squares) from one of the two common corners of the zones.*

The answer to an impossible question is always “no”, so we can assume Prover gets them for free, and she never asks such a question.

Definition 2.16 ***Forced*** *question is a question, which is not dummy, but the domino involved affects the current partial tiling.*

The answer clearly depends on the current tiling.

Definition 2.17 ***Open*** *question is a question, which is neither dummy nor forced.*

That is, an open question is about a domino:

1. The domino does not intersect the current (partial) cover.

2. It either is completely inside an *empty zone* or connects *two empty neighbouring zones.* In the latter case, the domino is also 12 squares away from any *zone corner.*

We should note that "dummy" is a static concept, i.e., it does not depend on the current partial tiling, whereas the concepts "forced" and "open"are dynamic.

We should also note that a forced/open question may be assigned to any of two neighbouring zones. Sometimes, we will need to assign a question to a particular zone. We will then use the following deterministic rules:

1. If the question is open, we associate it to either the *right* zone (if the border is vertical) or the *bottom* one (if the border is horizontal).
2. If the question is forced, we associate it to the zone, which has been covered *first*, starting from the last point in the game, when both zones were empty. In this way we ensure that the question was open to its zone at that time.

We can now describe

Reducer's algorithm

The randomised phase. For any zone, further called "current", we do the following: For any possible shape of any combination of its *nonempty* neighbours and any possible connection to its *empty* neighbours, either dent, hump or flat wall, we design a set of tilings of the current zone. These tilings have to have the property that not all of them agree on any open question associated to the current zone. It is very important to note, that the number of all the tilings is a *constant.* Adversary then chooses one of the tilings uniformly at random and remembers the choice through the entire Resolution Game. What remains to be proven is that a set of tilings, having the desired properties exists. This is proven in Appendix, lemma 2.21.

The deterministic phase. We should first note that at any round in Resolution Game, there are some "yes" and/or "no" dominoes on the mutilated chessboard. These are visible to both Prover Resolution and Reducer, who is also Adversary Resolution. Apart from them, there is a partial tiling of the board, which is visible only to Reducer. This tiling is consistent with all the "yes"/"no" dominoes until the end of the game, when Reducer gives up. In Tiling Game, there are some tiles on the board. Moreover, there is a correspondence between any tile and the corresponding zone in Resolution Game, as explained at the beginning of the section.

Let us now consider the four stages of the reduction for the two possible Prover Resolution's moves, either asking a question or forgetting:

1. Asking a question about a domino.

   (a) Prover Resolution asks a question in Resolution Game.

   i. The question is forced. Reducer answers immediately, without going to the next stage.
   ii. The question is open, and therefore it is the first one such question about the considered empty zone. Reducer switches from Adversary Resolution to Prover Tiling, thus going to the stage (b)

   (b) Prover Tiling asks a question in Tiling Game.

   (c) Adversary Tiling puts a tile on the board.

   (d) Given that tile and the neighbouring zones, Reducer gets the already chosen (in the randomised phase) cover of the zone and puts it on the mutilated chessboard, not revealing it to Resolution Prover. Reducer then switches back to Adversary Resolution and answers to Prover Resolution's question consistently with the cover just put.

2. Forgetting a domino, which is already on the mutilated chessboard.

   (a) Prover Resolution forgets a domino, already on the chessboard.
   i. The domino is not the only "yes"/"no" domino inside the corresponding zone. Reducer does not do anything else.
   ii. The domino is the only "yes"/"no" domino inside that zone. Reducer goes to stage (b).

   (b) Reducer, acting as Prover Tiling, removes the corresponding tile in the tiling game. After that, acting as Adversary Resolution, he forgets the cover of the zone, so that it becomes empty again.

The important lemma, that follows from our construction is the following:

Lemma 2.18 *At any round of Resolution Game, we can take a set of "yes"/"no" tiles, no two of them belonging to the same zone. They are independent and the probability that each of them agrees with the randomised phase is a constant, bounded away from both* 0 *and* 1*.*

Proof Because of the way we associate a question to a zone at the time when the zone was covered, the question was open for it. We can now use the fact, that the cover was chosen at random, among the set of covers, such that not all of them agree on any open question.□

That is enough to ensure that Theorem 2.10 applies, with the two colours, corresponding to the two possible answers to the chosen questions. The probabilities now are not 1/2 and 1/2, but some (small) constants, different from 0 and 1. This however does not affect the argument, so that the exponential lower bound is carried from Tiling Game through the reduction to Resolution Game.
Acknowledgements We would like to thank Mikhail Alekhnovich for finding the missing point already mentioned.

2.5 Appendix

Here, we shall give rigorous proofs of all the intuitively obvious, but tedious to prove, lemmas, used in the paper.

A lower bound on the length of a border, surrounding certain area. An arbitrary set of squares, R, further often called a *region* is given on the $m \times m$ board. We can then define the border and the complement of the region R, $\partial(R)$ and $co(R)$, as follows: $\partial(R)$ is the set of squares, having the property that each element in-there has a neighbour in R. $co(R)$ is all the rest, i.e. it contains every square that is in neither R nor $\partial(R)$. As usual, we also define the closure of R, $\overline{R} = R \cup \partial(R)$.

Clearly, we have $\partial(co(R)) \subseteq \partial(R)$. Note that the inclusion is not strict, as there can be squares in $\partial(R)$, surrounded only by squares in R, and therefore not in $\partial(co(R))$.

Lemma 2.19 *For any region R, $|\partial(R)|^2 \geq \min\left\{\left|\overline{R}\right|, \left|\overline{co(R)}\right|\right\}$.*

Proof We shall first prove a simple isoperymetric inequality.

Proposition 2.20 *Let C be a* connected region *of closure-area $\left|\overline{C}\right| = s$, that touches at most* two neighbouring sides *of the board. Then $\partial(C)$ has to be of length* at least $\sqrt{s}$ *(here both "area" and "length" mean "number of squares").*

Proof (of the proposition) W.l.o.g., we can assume that C, together with its border, is contained in an $a \times b$ rectangle, $a \geq b$. Obviously, the number of non-empty squares has to be at least a - at least one in every row of the rectangle, as no row can be bounded by two opposite sides of the board. Then

$$a^2 \geq ab \geq s.$$

□

Note that this proposition does not hold if C touches *three* of the sides of the chessboard. As an example, we can take small number of squares connecting two neighbouring sides of the board, near to one of the corners. They divide it into two connected areas, one of them being much smaller than the other. It is now clear that the proposition does not hold for the bigger component.

We can now prove the lemma. Let us "transform" R as follows: We first consider all *connected* sub-regions of R. They are disjoint, but their borders are not, in general. We then join two neighbouring sub-regions, having intersecting borders. In doing this, we could need to make some of the common-border squares internal to the union. Therefore the overall area increases, whereas the overall border-length decreases. We repeat the above procedure while possible. In the end, we have a set of connected regions, $R_1, R_2, \ldots R_k$, such that

[(a)]They are disjoint, and so are their borders, $\partial(R_1), \partial(R_2), \ldots \partial(R_k)$. These imply

$$\left|\overline{R_1}\right| + \left|\overline{R_2}\right| + \ldots \left|\overline{R_k}\right| = \left|\overline{R}\right|$$

$|R_1| + |R_2| + \ldots |R_k| \geq |R|$

(b) $|\partial(R_1)| + |\partial(R_2)| + \dots |\partial(R_k)| \leq |\partial(R)|$

We finish the proof by a case-analysis:

1. Each R_j touches two neighbouring sides of the chessboard at most. The proposition 2.20 then applies to all the R_js. Those inequalities, together with (a) and (c) above, give

$$|\partial(R)|^2 \geq (|\partial(R_1)| + |\partial(R_2)| + \dots |\partial(R_k)|)^2 \geq$$
$$|\partial(R_1)|^2 + |\partial(R_2)|^2 + \dots |\partial(R_k)|^2 \geq$$
$$|\overline{R_1}| + |\overline{R_2}| + \dots |\overline{R_k}| = |\overline{R}|,$$

as desired.

2. There is at least one R_j, that touches either two opposite sides of the board or three sides of the board. In both cases, there has to be a "path" of border squares connecting two opposite sides of the chessboard. Every such a path contains at least m squares, and we are done.

3. There is at least one R_j, that touches all four sides of the chessboard. We now consider the intersections of such an R_j with every horizontal line. If each such intersection contains at least one border point, there are at least m border squares, and we are done. If not, there is a row, consisting of only interior points of R_j. It this case, we need to consider $co(R)$. All what we have done so far with R applies to $co(R)$, too. Let us however observe that only the previous two cases, 1 and 2 are now possible, as there is entire horizontal line in R, that is *not* in $\overline{co(R)}$, dividing the board into two disjoint rectangles. Thus

$$|\partial(R)|^2 \geq |\partial(co(R))|^2 \geq \left|\overline{co(R)}\right|.$$

The first inequality is because of $\partial(co(R)) \subseteq \partial(R)$. This completes the proof.

□

Let us note that the strongest, sharp, version of the above lemma is as follows: Consider an $m \times m$ board, which is divided into three disjoint subset R, S and T. Suppose also that S separates R from T, i.e. there are no two neighbouring squares such that one is in R and the other is in T. The following inequality holds:

$$|S|^2 - |S| \geq 2\min\{|R|, |T|\}.$$

Possibility of tiling zones of certain shape, contained in big, 48×48, squares. We first need to remind what the *shape* of a *zone* is. A *zone* is, roughly speaking, a big, 48×48, square, with "few" small squares cut off. We shall first explain how the missing squares can exactly appear.

We first divide a zone into 12×12 smaller, 4×4, squares, further called *sub-zones*, as done on figure 2.9. *Missing squares* can only appear on the *sides* of a zone, inside the four dashed "bands" which are the border part of the four gray five-sub-zone areas (figure 2.9A). Moreover, there are only the following possible shapes:

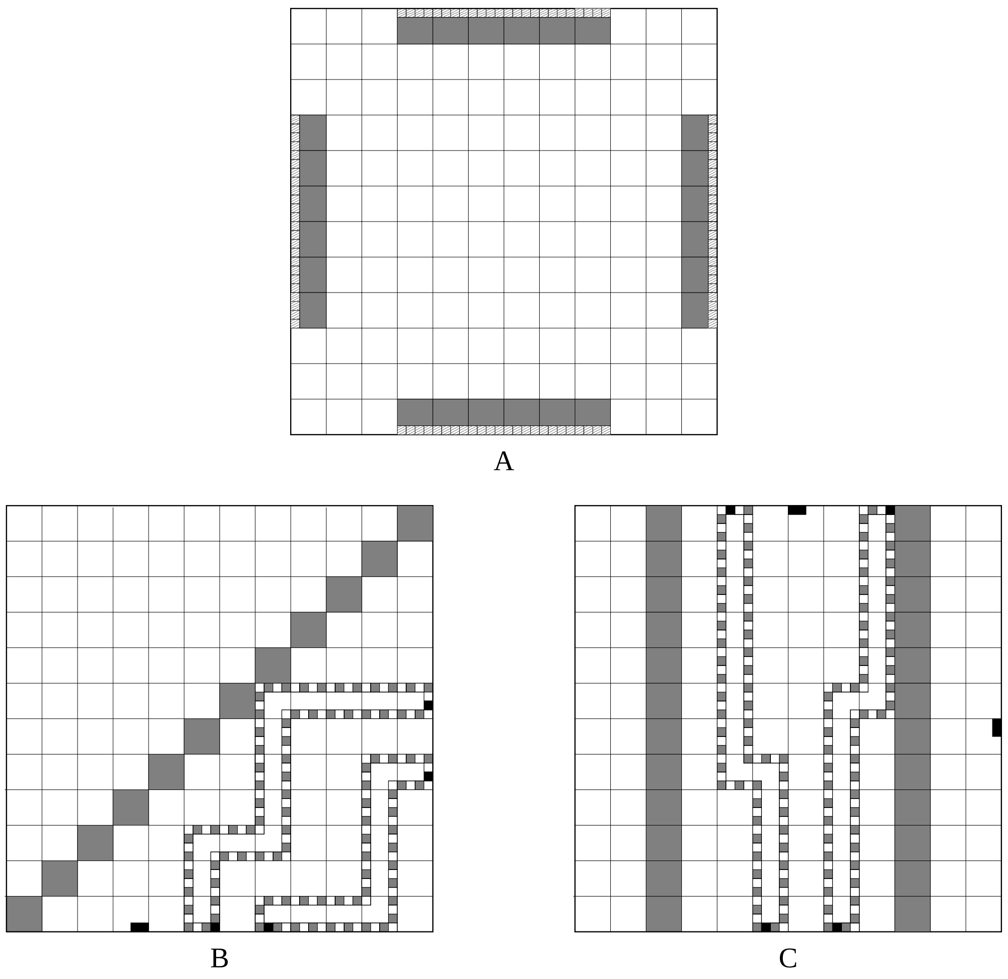


Figure 2.9: Zones

1. Two neighbours , of *different colour*, belonging to a sub-zone, and *not* being the two *middle* squares. An example is given on the eastern border of the zone C, figure 2.9 (here and everywhere on figure 2.9, the missing squares appear in black). Remember, in the tiling game, this corresponds to a *flat wall*, as well as the case, where no missing squares appear (the western side of the zone C, as an example).

2. Two squares of *the same colour,* belonging to sub-zones, that are a sub-zone away from each other. The eastern border of the zone B is an example. There are *two black* squares missing, which corresponds to a *hump* in the tiling game (*two white* squares would correspond to a *dent*).

3. Four missing squares, that are a *combination* of the previous two cases, and, moreover, no two mutilated sub-zones can be neighbours. The southern border of zones B and the northern border of zone C are such examples. Again, if *two more white* squares than black ones are missing, this is *a dent* in the tiling game (B). The symmetric case (C) corresponds to *a hump.*

We should not however forget that we have an additional domino, which corresponds to the open question. Thus we define a *mutilated zone* to be a zone with a missing domino. Moreover, the number of the white squares, *present* in the zone has to be equal to the number of white ones. That is, in the tiling game, the number of dents is equal to the number of humps. This completes the description of what a mutilated zone is.

As mentioned in the paper, the proposition we need is

Lemma 2.21 Any mutilated zone *can be covered by dominoes.*

Proof For a moment, we will ignore the last domino, cut off from the zone. That is, we will first show how to deal with the missing border squares only. We can then “repair” the solution in order to tackle the domino, we have “forgotten” about.

In doing the first part, we proceed by a very simple case analysis.

- There are a “dent” and a “hump” on *two neighbouring* sides of the zone, as shown on figure 2.9B. We “connect” the dent and the hump by two *roads*, as shown on the same picture. The missing squares “vanish”, “absorbed” by the roads (the squares of the roads are shown in full detail, i.e. colour; the missing squares are in black, as explained before). The only missing squares that are not absorbed, are those from the first case above. It is important to observe that the two roads can be designed so that they do not touch each other and none of them crosses one of the diagonals (in our case, the gray diagonal). If the other two neighbouring sides contain a dent and a hump, we connect them in a similar way.

- There are a dent and a hump on *two opposite* sides of the zone and the other two sides are flat walls. We connect the dent and the hump, as shown on figure 2.9C. Again, the two road do not touch each other and are bounded by the two gray (vertical, in our case) lines.

- All the sides are flat walls. We do not do anything for a flat wall. As explained before, any two missing neighbours remain.

We can now “cut off” the last domino. In general, it affects some road. What we need, in order to complete the proof, is to show how to “repair” the affected road. We first observe that only one road may need to be repaired. This is due to the clever design of the roads, that ensures, no two of the touch each other. We need now to consider all the possible ways, the domino can be “put” inside the zone with already created roads. They are shown on figure 2.10. All possible *sub-zones* are shown on the first row, where the part(s) of a road are dashed (for convenience, we think of any two missing neighbours as a “small” road). The rest of the figure shows how to repair the road *locally.* The cut off domino is the bold-face bordered one. The dashed border dominoes are part of a road. The gray squares are missing squares (note that they can only appear in an end sub-zone). The normal bordered dominoes are the new introduced ones, that is the ones, needed in “repairing” of the road. The possible cases are:

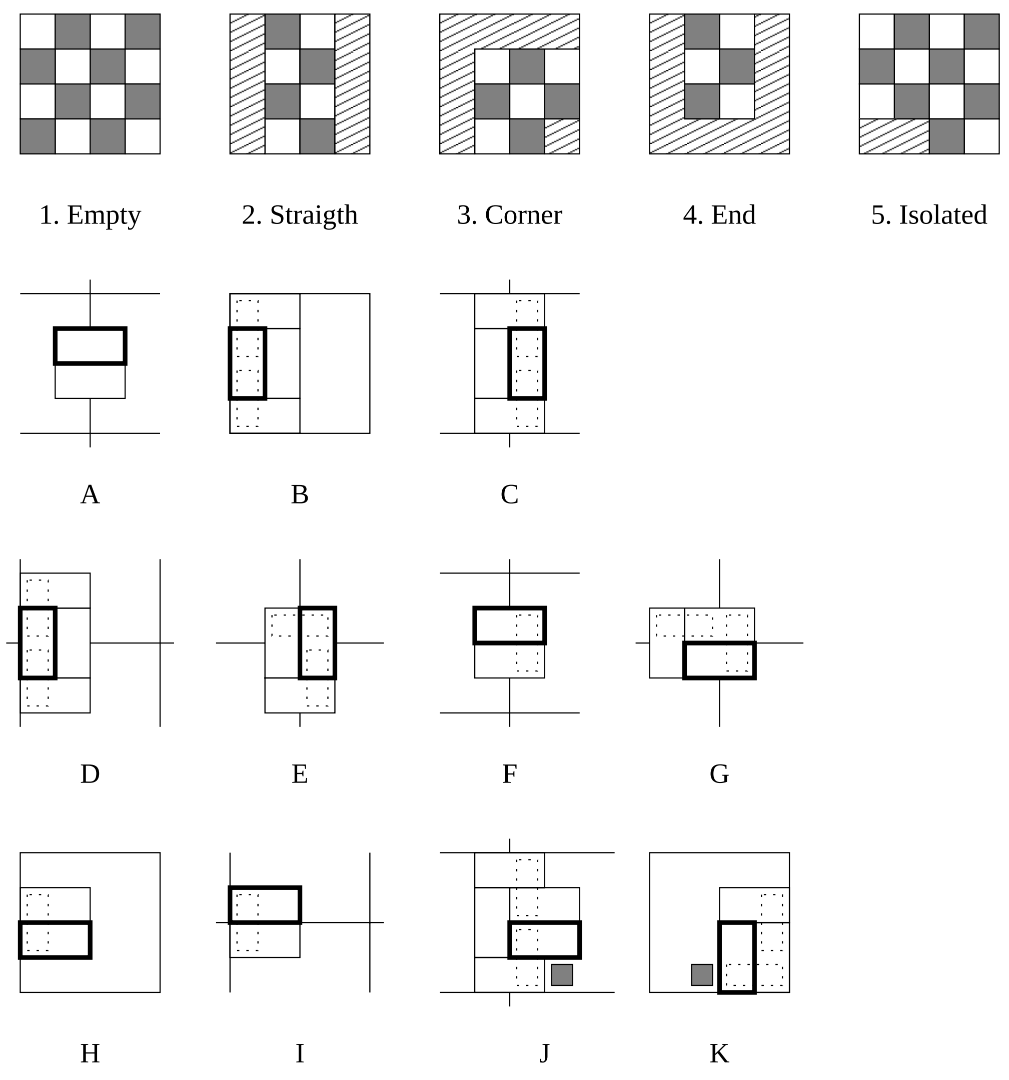


Figure 2.10: Repairing the road

1. The simplest one, when the domino is inside a sub-zone and does not affect a road. It is trivial to check, by looking at the pictures 1-5 (note 5 is not a part of a real road, as explained before) , that the corresponding sub-zone can be tiled by dominoes.

2. The domino affects two neighbours, but does not affect a road, as shown on figure 2.10A. This clearly reduces to two instances of the previous case.

3. The domino lies on the road, inside a sub-zone. This can be fixed, as shown on B and C . It is not hard to see, that each of them may be impossible, so that both are needed to be considered.

4. The domino lies on the road, affecting two sub-zones. This reduces to at most two instances of the first case, as shown on D and E.

5. The domino cuts only one square from the road, affecting two neighbouring

sub-zones. F or G reduces it again to the first case.

6. The domino cuts only one square from the road, and it is inside a sub-zone. This is the most complicated case, because of the two possibilities for the affected road domino (H and I) and also because of the end sub-zones (J and K). It is however not hard to see that these four cases are all the possibilities.

This completes the proof. □

Chapter 3

Tree-Resolution Complexity of the Weak Pigeon-Hole Principle

We show some tight results about the tree-resolution complexity of the Weak Pigeon-Hole Principle, PHP_n^m. We prove that any optimal tree-resolution proof of PHP_n^m is of size $2^{\theta(n \log n)}$, independently from m, even if it is infinity. So far, the lower bound know has been $2^{\Omega(n)}$. We also show that any, not necessarily optimal, regular tree-resolution proof PHP_n^m is bounded by $2^{O(n \log m)}$. To best of our knowledge, this is the first time, worst-case proofs have been considered, and a non-trivial upper bound has been proven. Finally, we discuss and conjecture some refinements of Riis' Complexity Gap theorem for tree-resolution complexity of Second Order Existential ($SO\exists$) sentences of predicate logic.

3.1 Introduction

The Pigeon-Hole Principle (PHP) is probably the simplest and at the same time the most widely used combinatorial principle. In its classical formulations, it states that there is no *injective* map from a finite m-element set to a finite n-element set if $m > n$. PHP_n^m is very intuitive for the human way of thinking, and it is also easily provable within set theory. This is however not the case for some *propositional proof systems.* In his seminal paper [31], Haken showed that *any resolution proof* of PHP_n^{n+1} is of size $2^{\Omega(n)}$. His proof has been generalised and simplified in [16], [7], [11]. For quite a while, the best known result had been a $2^{\Omega\left(n^2/m\right)}$ lower bound from [16], thus having left the case $m = \Omega\left(n^2/\log n\right)$ as an important open problem in resolution proof complexity. An important step has been done in [38], where a $2^{\Omega(n^{\varepsilon})}$ lower bound on *any regular-resolution proof* of PHP_n^m is proven. Shortly afterwards, the problem has finally been solved by Raz in [42], and further strengthen and improved by Razborov in [43], [44].

In this paper, we concentrate on *tree-resolution.* Even though it is probably the weakest propositional proof system one could think of, the exact complexity of tree-resolution proofs of PHP_n^m has not been known so far. A $2^{\Omega(n)}$ lower bound was shown in [15], whereas one can construct only a $2^{O(n \log n)}$ tree proof by "unfolding" the $2^{O(n)}$ DAG-resolution proof given in the same paper. A $2^{O(n \log n)}$ lower was proven in [32], but only for *ordinary pigeon-hole principle*, i.e. PHP_n^{n+1}.

The first contribution (section 3.3) of our paper is closing the gap. We prove a $2^{\Omega(n \log n)}$ *lower bound* on *any tree-resolution proof* of PHP_n^m, independently from m, even if it is infinity. It is *tight* up to a constant factor in the exponent or, in other words, up to *a polynomial transformation.* In order to prove the result, we introduce a new method for proving lower bounds on tree-resolution proofs. It is more general than the existing one (see e.g. [48]). The latter works only for balanced proofs, whereas any tree-like resolution proof of PHP_n^m is highly *unbalanced* as shown in the paper.

The second contribution (section 3.4) of the paper is considering *the worst tree regular resolution* proofs of PHP_n^m. To best of our knowledge, this is for the first time, *the worst case* proof complexity is considered. We prove an *upper bound* of $2^{O(n \log m)}$, which is non-trivial, as there are mn variables, and one can therefore expect the worst case to be as bad as 2^{mn} (we consider of course only proofs which do not contain vacuous weakening of axioms). This has the following very interesting consequence: Consider $PHP_n^{poly(n)}$, i.e. m is some polynomial in n. The optimal and the worst regular tree-resolution proofs of $PHP_n^{poly(n)}$ are *polynomially related*, and so are any two *random* regular tree-resolution proofs. This also has an implication in automated theorem proving, as it shows that there are natural problems for which any *DLL-based* proof search heuristic is as good as any other.

Finally (section 3.5), we discuss and conjecture some refinements of Riis' Complexity Gap theorem for tree-resolution complexity of Second Order Existential ($SO\exists$) sentences of predicate logic [48], motivated by our results. These conjectures nicely relate tree-resolution gap(s) to a possible general-resolution gap as well as to a characterisation, involving optimal and worst-case tree-resolution proofs.

3.2 Preliminaries

We first give some definitions. A *literal* is either a propositional variable or the negation of a propositional variable. A *clause* is a set of literals. It is satisfied by a truth assignment if at least one of its literals is true under this assignment. A set of clauses is *satisfiable* if there exists a truth assignment satisfying all the clauses.

As we have already said, by PHP_n^m we denote the claim that there is no *injective map* from a set of size m to a set of size n, where $m > n$. We encode its negation as the following set of clauses

1. $\{p_{i1}, p_{i2}, \dots p_{in}\}$ for $1 \leq i \leq m$
2. $\{\overline{p}_{ij}, \overline{p}_{ik}\}$ for $1 \leq i \leq m$, $1 \leq j < k \leq n$

We allow m to be infinity. In this case, we have an infinite set of clauses, but all the clauses themselves are finite. Although we consider the *injective* PHP, all the results and proofs from the paper remain valid for the *bijective* PHP, too.

Resolution is a proof system designed to *refute* given set of clauses i.e. to prove that it is unsatisfiable. This is done by means of the resolution rule

$$\frac{C_1 \bigcup \{v\} \quad C_1 \bigcup \{\overline{v}\}}{C_1 \bigcup C_2}.$$

Thus, we can derive a new clause from two other clauses that contain a variable and its negation respectively. The goal is to derive the empty clause from the initial ones. Anywhere we say we *prove* some proposition, we mean that first we take its negation in a clausal form and then resolution is used to refute these clauses.

There is an obvious way to represent every resolution refutation as a directed acyclic graph whose nodes are labelled by clauses. The sources, i.e. the vertices with no incoming edges, are the initial clauses. The only sink, i.e. the vertex with no outgoing edges, is the empty clause. Everywhere in the paper, we say *"the size of a proof"*, we really mean *the number of vertices* in the corresponding graph.

We can now define two important restricted versions of resolution. First one is *tree resolution* when the graph is a tree or, in other words, we are not allowed to reuse any previously derived clauses. The other one is r*egular resolution* when every variable is resolved at most once along any path from a source to the sink.

For an unsatisfiable set of clause, we can consider the following *search problem*: given a truth assignment, find a clause which is falsified under it. There is a close connection between refuting an unsatisfiable set of clauses by some proof system and solving the corresponding search problem within some model of computation. In [33], it is proven that tree-resolution refutations are equivalent to *boolean decision trees.* More precisely, given a refutation of the set of clauses, it can be viewed as a decision tree, solving the search problem and vice versa. The same result holds for regular resolution refutations and *read-once branching programs.* In contrast to these, general resolution proofs are not equivalent to branching programs. As a matter of fact, there is a polynomial-size branching program, solving the search problem corresponding to PHP_n^{n+1} while all resolution refutations are of exponential size.

Everywhere in the paper, we use the equivalence between a tree-resolution proof and a boolean decision tree. All the proofs are, in fact, for decision trees, whereas the results are stated in terms of tree-resolution proofs. We only consider proofs that are regular. This is not a restriction at all as in a decision tree, it does not make sense to query any variable more than once. On the other hand, if we do not set this restriction, we would not be able to prove any upper bounds, as any given proof can be extended by (unbounded) number of "meaningless" applications of the resolution rule. Thus, from now on, every time we say "tree resolution", we really mean "regular tree resolution". As already mentioned we do not allow proofs to contain vacuous weakening of axioms. In terms of decision trees a branch terminates as soon as a contradiction is reached.

An important technique, we use to prove lower bounds on proofs, is considering a proof as a *Prover-Adversary game.* It is first introduced in [41] and developed further in [40] for general resolution. For tree resolution, it can be simplified, as done in [48]. Adversary claims that there is a satisfying assignment. Prover's task is to expose him. In order to do that, Prover asks questions about variables according to a decision tree, she holds. Clearly, there is no way for Adversary to win the game. His task is therefore to enforce a big enough subtree, contained in Prover's decision tree. If he has a strategy, enforcing that, no matter what strategy Prover uses, we have a lower bound on the tree-resolution refutations of the given set of clauses.

The only Adversary's strategy, used so far, essentially shows that there are certain number of branching points in any decision tree. It implies the existence of *a big balanced subtree* of a certain hight, thus proving an exponential in the hight lower bound. Unfortunately, this technique does not work for *unbalanced decision trees.* PHP^m_n tree-resolution refutations is such an example as we shall see in the next section. In order to tackle these, we introduce new, more general method for proving lower bounds. It requires defining a function on the nodes of the decision tree. The value of the function at any node should be a lower bound of the size of the subtree rooted by that node. Thus the function value on the root is a lower bound on the size of any decision tree solving the given search problem.

3.3 Optimal proofs

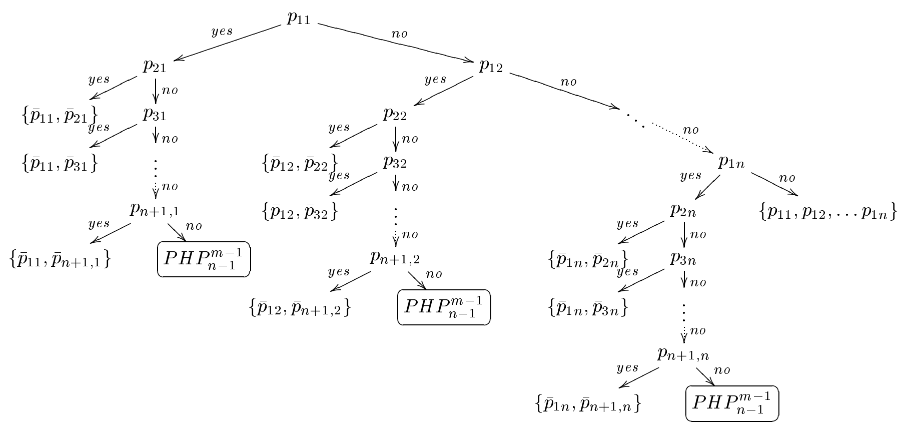
We first construct a $2^{O(n \log n)}$ tree-resolution proof (in fact, boolean decision tree, as we have already mentioned), and we prove the corresponding lower bound.

Here we fix some notations that we will use in both this and the next section. We denote the bigger, m-element set by M, and the other, n-element set by N. We consider M and N as the two parts of the complete bipartite graph $K_{m,n}$, and then there is 1-1 correspondence between the edges of the graph and variables p. Thus we can speak about a partial matching in $K_{m,n}$ instead of a partial function form M to N. All the queries/questions, from the decision tree, are about the edges. We can however say that a question is about a vertex, too if the corresponding edge is incident to that vertex.

Upper bound

The sketch of the construction is as follows. Obviously, Prover can restrict herself to the first $n+1$ elements of M. She asks consecutively all the questions about the first element from M, namely p_{11}, p_{12}, $\dots$ p_{1n}. If all the answers are "no", a contradiction is found. Otherwise, suppose p_{1j} is the first question with a positive answer. Prover then asks all the remaining questions about the j-th element of N, namely p_{2j}, p_{3j}, $\dots$ $p_{n+1,n}$. If at least one answer is "yes", a contradiction is found. If not, we can safely remove the first element from M and the j-th element from N, and then look for a contradiction on a PHP^{m-1}_{n-1} instance.

The boolean decision tree is given on the figure 3.1 below. The internal nodes are labelled with the queried variables, and the edges are marked with the corresponding answer. Every external node (leave) is labelled by the found



p_{11}
yes
no
p_{21}
p_{12}
$\{\bar{p}_{11}, \bar{p}_{21}\}$
p_{31}
$\{\bar{p}_{11}, \bar{p}_{31}\}$
$p_{n+1,1}$
$\{\bar{p}_{11}, \bar{p}_{n+1,1}\}$
PHP^{m-1}_{n-1}
p_{22}
$\{\bar{p}_{12}, \bar{p}_{22}\}$
p_{32}
$\{\bar{p}_{12}, \bar{p}_{32}\}$
$p_{n+1,2}$
$\{\bar{p}_{12}, \bar{p}_{n+1,2}\}$
PHP^{m-1}_{n-1}
p_{1n}
$\{p_{11}, p_{12}, \dots p_{1n}\}$
p_{2n}
$\{\bar{p}_{1n}, \bar{p}_{2n}\}$
p_{3n}
$\{\bar{p}_{1n}, \bar{p}_{3n}\}$
$p_{n+1,n}$
$\{\bar{p}_{1n}, \bar{p}_{n+1,n}\}$
PHP^{m-1}_{n-1}

Figure 3.1: An optimal decision tree for PHP^m_n

contradiction, i.e. a clause falsified under the (partial) truth assignment corresponding to the path from the root to this node. The nodes marked by PHP^{m-1}_{n-1} are, in fact, subtrees.

What remains is to estimate the size. The decision tree for PHP^m_n consists of n copies of the decision tree for PHP^{m-1}_{n-1} plus a quadratic in n overhead. More precisely

$$S(n) = \begin{cases} nS(n-1) + 2n^2 + n + 1 & \text{if } n > 1 \\ \quad 5 & \text{if } n = 1 \end{cases},$$

where $S(n)$ is the size of the decision tree for PHP^m_n.

It is now easy to prove by induction that $S(n) \leq 6(n+1)!$. Finally, an application of Stirling's approximation of the factorial gives the desired upper bound.

Lower Bound

The main idea in our proof is to define a function on the nodes of the decision tree. The value of the function at any node should be a lower bound of the size of the subtree rooted by that node. After having done that, it suffices to compute the function value on the root. The result is a lower bound on the size of any decision tree, solving the search problem for PHP^m_n.

We assume, w.l.o.g., that n is even. W.l.o.g. we can also assume that Prover's decision tree is *read-once*, i.e. along every path any question is asked at most once. Now, we can explain Adversary's strategy.

An important concept, we introduce here, are *counters.* A counter is attached to every vertex in M which is not matched yet to any vertex in N. In addition, there is one special counter that will be explained later on. Initially all the counters are set to zero. During the game, every counter is an upper bound of the number of vertices in N that are "forbidden" for the corresponding vertex in M. When some counter reaches the value n, Adversary "gives up", although it might be possible to continue the game a few more rounds.

We can now classify all the questions that can appear in the decision tree and show how to maintain the counters. Let k be the size of the partial matching obtained so far, i.e. the number of "yes" answers along the path from the root to the current node. There are three kinds of queries:

1. *Free-choice.* Neither of the two vertices involved is in the current partial matching and the counter of the vertex from M is less than $\frac{n}{2}+k$. Adversary chooses either "yes" or "no" answer with some probability. The actual probability does not matter, the important point is that the free choice forces Prover to branch the decision tree at that point. If the answer is "no", only the counter of the element form M increases by one. If the answer is "yes" this counter is cancelled, i.e. not maintained any more, but the counters of all the other elements in M are increased by one.

2. *Critical.* Neither of the two vertices involved is in the current partial matching but the counter of the vertex from M is equal to $\frac{n}{2}+k$. Adversary answers "yes", he current counter is cancelled, and the counters of all the other elements in M are increased by one.

3. *Forced.* Some of the vertices involved (or both) is already in the matching. Adversary answers "no" and does not change any of the counters attached to the elements in M. He however increases by one the special counter, which counts the forced questions.

First of all, it is easy to see that for a given element in M, its counter is an upper bound on the number of elements in N that cannot be matched to that element. There are also some other simple observations to be made. First one says that Adversary always "survives" certain number of rounds.

Lemma 3.1 *A contradiction can be found only when some counter reaches the value n. In this case, at least $\frac{n}{2}$ "yes" answers must be present on the path from the root to the current node.*

Proof A simple induction on k proves the following assertion: All the counters are bounded from above by $\frac{n}{2}+k$ along any path from a node, where the partial matching is of size k, to the node, where that size becomes $k+1$. The lemma then follows.□

The next lemma shows that there must be a very long branch in any decision tree. Together with the main result, it implies that every such tree is unbalanced.

Lemma 3.2 *In every decision tree for PHP^m_n, there is a path of length $\Omega\left(n^2\right)$.*

Proof Consider the path, where Adversary answers "no" to every free-choice question. It is now easy to observe that when k-th critical questions asked, the corresponding vertex from M has a counter value equal to $\frac{n}{2}+k-1$. That counter has been increased $k-1$ times because of the previous $k-1$ critical question. The remaining $\frac{n}{2}$ increases are result of "no" answers to free-choice question about the corresponding vertex. Thus, along the particular path, we consider, any “yes” answer is preceded by $\frac{n}{2}$ negative answers about the same vertex.

The lemma 3.1 claims that every path contains at least $\frac{n}{2}$ “yes” answers. Therefore our path contains at least $\frac{n^2}{4}$ “no” answers.□

We can now prove the main result.

Theorem 3.3 *Every tree-resolution proof of PHP_n^m is of size $2^{\Omega(n \log n)}$.*

Proof First we define an appropriate function as it has been explained in the beginning of the section.

Let us denote by k the size of the partial matching at the current node u, i.e. the number of “yes” answers along the path from the root to u. Let us also sort the $m-k$ unmatched vertices from M in decreasing order of their counters, and denote the values of the counters themselves by $p_1 \geq p_2 \geq \ldots \geq p_{m-k}$. The forced question counter is denoted by p_0. The value of the function at the node is then defined by

$$f(u) = \prod_{i=1}^{\frac{n}{2}-k} q_i, \qquad \text{where} \quad q_i = \begin{cases} \frac{n}{2} + k - i - p_i & \text{if it is positive} \\ 1 & \text{elsewhere} \end{cases}$$

On the root, r, we have $f(r) = \left(\frac{n}{2} - 1\right)!$, so that $f(r) = 2^{\Omega(n \log n)}$. It only remains to prove that at any node the function value is a lower bound for the size of the subtree rooted by the node.

The proof is by induction on the tuples of the form

$$\left(p_1, p_2, \ldots, p_{\frac{n}{2}-k}, p_0 + \sum_{i=\frac{n}{2}-k+1}^{m-k} p_i \right).$$

We order them as follows. The shorter a tuple, the smaller it is. If two tuples have equal length, the lexicographically *bigger* one is the smaller. Clearly, this ordering makes the induction work from the leaves to the root of the decision tree, as the tuple on any node is strictly bigger than the tuples on its successors in the tree.

The basis case is then $k = \frac{n}{2}$, where $f(u) = 1$, as the product is empty. Obviously, the function value at the node is a lower bound of the corresponding subtree, no matter what the only element of the tuple is.

To prove the induction steep, we need to consider all possible kind of questions that can appear at the current node u.

1. Forced. We consider the “no” branch only. Denoting its root (the “no” successor of u) by v, we have $f(u) = f(v)$, as only p_0 increases by one when going from u to v and f does not depend from p_0. By the induction hypothesis, we are done.

2. Critical. W.l.o.g. we assume that the question is about the element, having p_1 as a counter. It is so, because a critical question always involves the biggest counter (Even if there are many counters with the biggest

value $\frac{n}{2}+k$, we can always consider p_1, as two elements, having the same counter value are indistinguishable to Adversary's strategy). We consider the "yes" branch only. Denoting the "yes" successor of u by v, we have again $f(u)=f(v)$. That is the case, because all the counters $p_2,\ldots,p_{\frac{n}{2}-k}$ increase by one when going from u to v, but so does k, therefore the contributions $q_2,\ldots,q_{\frac{n}{2}-k}$ do not change. q_1 vanishes at v, but its value at u is one, as $p_1=\frac{n}{2}+k$. By the induction hypothesis, we are done.

3. Free-choice. There are three sub-cases:

   (a) The index involved, j, is greater than $\frac{n}{2}-k$. W.l.o.g. we can also assume $p_{\frac{n}{2}-k} > p_j$ since if they were equal Adversary could behave as the question were about $\frac{n}{2}-k$-th element (again, any two vertices having the same counter value are indistinguishable to Adversary's strategy). The "no" answer then does not change anything except the last element of the tuple, but f does not depend on it. So, $f(u)=f(v)$, where v is the "no" successor of u. By the induction hypothesis, we are done.

   (b) The index involved, j, is between 1 and $\frac{n}{2}-k$, but the contribution, q_j, of that element to the function f is one. That is similar to the previous sub-case, as the "no" answer leaves the value of f unchanged when going from from u to to its "no" successor v.

   (c) The index j is between 1 and $\frac{n}{2}-k$ and the contribution, q_j, of that element to the function f is greater than one. This is the only non-trivial case, in the sense that we need consider both subtrees of the current node u. Note that if there are many counters, having the same value equal to p_j, w.l.o.g. we can think that j is the minimum such index, so that the "no" answer does not change the order of the counters.
   The "no" subtree gives the tuple

$$\left(p_1,\ldots p_{j-1},p_j+1,p_{j+1}\ldots,p_{\frac{n}{2}-k},p_0+\sum_{i=\frac{n}{2}-k+1}^{m-k} p_i\right)$$

   and the value

$$f(v)=(q_j-1)\prod_{\substack{i=1\\ i\neq j}}^{\frac{n}{2}-k} q_i.$$

   The "yes" subtree gives

$$\left(p_1+1,\ldots p_{j-1}+1,p_{j+1}+1\ldots,p_{\frac{n}{2}-k}+1,m-\frac{n}{2}+p_0+\sum_{i=\frac{n}{2}-k+1}^{m-k} p_i\right)$$

and the value

$$f(w) = \prod_{\substack{i=1 \\ i \neq j}}^{\frac{n}{2}-k} q_i.$$

The induction hypothesis then applies to both subtrees, so the size of the current subtree is at least

$$1 + f(v) + f(w) = 1 + f(u) > f(u).$$

This completes the proof.□

3.4 Worst case proofs

We first construct a $2^{O(n \log m)}$ boolean decision tree for PHP_n^m which is a lower bound for the worst-case regular tree -resolution proofs. We also show the same upper bound, i.e. any such proof cannot be worse than that. It is very important to now note that "worst case", in our context, has *a completely different meaning* than the usual one, used in Complexity Theory or Analysis of Algorithms.

Lower bound

The sketch of the construction is as follows. Prover ask all the questions about the first element from N, namely p_{11}, p_{21}, ... p_{m1}. If all the answers are "no", we can remove the first element from N, and thus get an PHP_{n-1}^minstance. Otherwise, suppose p_{i1} is the first question with a positive answer. Prover then asks all the remaining questions about the first element of N, namely p_{i+11}, p_{i+21}, ... p_{m1}. If at least one answer is "yes", a contradiction is found. If not, we can safely remove the first element from N and the i-th element from M, and then look for a contradiction on a PHP_{n-1}^{m-1} instance.

The boolean decision tree is given on the figure 3.2 below.

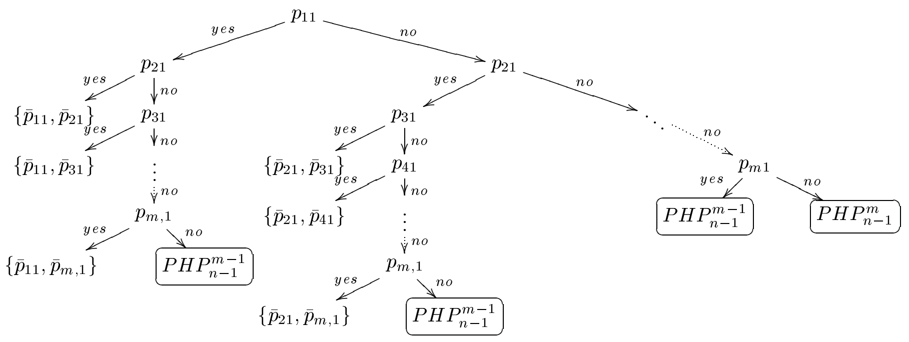

p_{11}
yes
no
p_{21}
p_{21}
$\{\bar{p}_{11}, \bar{p}_{21}\}$
p_{31}
$\{\bar{p}_{11}, \bar{p}_{31}\}$
$p_{m,1}$
$\{\bar{p}_{11}, \bar{p}_{m,1}\}$
PHP_{n-1}^{m-1}
p_{31}
$\{\bar{p}_{21}, \bar{p}_{31}\}$
p_{41}
$\{\bar{p}_{21}, \bar{p}_{41}\}$
$p_{m,1}$
$\{\bar{p}_{21}, \bar{p}_{m,1}\}$
PHP_{n-1}^{m-1}
p_{m1}
PHP_{n-1}^{m-1}
PHP_{n-1}^{m}

Figure 3.2: A worst-case decision tree for PHP_n^m

What remains is to estimate the size. The decision tree for PHP_n^m consists of m copies of the decision tree for PHP_{n-1}^{m-1}, one decision tree for PHP_{n-1}^m

plus a quadratic in m overhead. More precisely

$$S(m,n) = \begin{cases} mS(m-1,n-1) + S(m,n-1) + m^2 & \text{if } n > 1 \\ 5 & \text{if } n = 1 \end{cases},$$

where $S(m,n)$ denotes the size of the decision tree for PHP_n^m.

We have

$$S(m,n) > mS(m-1,n-1) > m(m-1)S(m-2,n-2) >$$

$$\ldots \prod_{i=0}^{\lceil \frac{n}{2} \rceil - 1} (m-i)\, S\left(m - \left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right).$$

Therefore, for every $m > n > 2$, we get

$$S(m,n) > 5\left(m - \left\lceil \frac{n}{2} \right\rceil\right)^{\lceil \frac{n}{2} \rceil} = 2^{\Omega(n \log m)}.$$

Upper bound

The main idea is the same as in the proof of the lower bound on the optimal refutation. This time however, we introduce the counters to the elements of the set N. Every counter p_j equals to m minus the number of questions about the j-th element of N that have already been asked. In other words, the counter contains *exactly* the number of possible questions about the element to be asked in the future. There is also one global counter p_0 that is the sum of all the counters p_j, $1 \leq j \leq n$.

We can now prove the main result of this section.

Theorem 3.4 *Every regular tree-resolution proof of PHP_n^m is of size $2^{O(n \log m)}$.*

Proof Again we define an appropriate function on the nodes of the read-once decision tree. At any node the value of the function will be an upper bound on the size of the subtree rooted at that node.

Let us denote by u the current node, and by P, $P \subseteq N$, the set of all the vertices from N that are not yet matched to any vertex in M. The function f is the defined as

$$f(u) = 2(p_0 + 1) \prod_{j \in P} (p_j + 1) - 1.$$

At the root of the tree, r, we have $f(r) = 2(mn+1)(m+1)^n - 1$, so that $f(r) = 2^{O(n \log m)}$. It only remains to prove that at any node the function value is an upper bound for the size of the subtree rooted by the node.

The proof is by induction on the global counter p_0 .

The basis case is then $p_0 = 0$, so that all other p's are zeros and therefore $f(u) = 1$. In this case all variables have already been queried, as there are no possible questions left. Therefore a contradiction has already been found and $f(u) = 1$ is an upper bound.

To prove the induction steep, we consider the following two cases.

1. The question at the current node, u, is about the i-th element from N, and $i \notin P$. This means that element has already been matched to some element in M, so that the current question is forced. Therefore, the "yes" subtree consists of a single vertex, labelled by the contradiction found. Let us denote by v the "no" successor of u. The induction hypothesis applies at v, as p_0 decreases by one there, so the size of any subtree rooted at u is at most

$$2 + f(v) = 2 + 2p_0 \prod_{j \in P} (p_j + 1) - 1 \leq 2(p_0 + 1) \prod_{j \in P} (p_j + 1) - 1 = f(u).$$

2. The question at the current node, u, is about the i-th element from N, and $i \in P$. The induction hypothesis then applies to both "yes" and "no" successors of u. Denoting them by v and w respectively, we have that the size of any subtree rooted at u is at most

$$1 + f(v) + f(w) =$$
$$1 + 2p_0 \prod_{j \in P \setminus \{i\}} (p_j + 1) - 1 + 2p_0 p_i \prod_{j \in P \setminus \{i\}} (p_j + 1) - 1 =$$
$$2p_0 \prod_{j \in P} (p_j + 1) - 1 < 2(p_0 + 1) \prod_{j \in P} (p_j + 1) - 1 = f(u).$$

This completes the proof.□

3.5 Link to Complexity Gap theorem

In this section, we discuss a possible strengthening of Riis' complexity gap theorem for tree resolution. We first state the theorem and conjecture that it can be extended to show a gap not only between $\theta(\text{poly}(n))$ and $2^{\theta(n)}$ but also from $2^{\theta(n)}$ to $2^{\theta(n \log n)}$. We also conjecture the existence of a gap for *general resolution* and its connection with the gap for tree resolution. Let us also mention that there is no complexity gap above $2^{\theta(n \log n)}$, and, moreover, there are uniform, i.e. $SO\exists$-generated, tautologies having highly non-uniform, *fluctuating*, tree-resolution refutations. The proofs of these are however not included in the present paper as they are out of its scope.

Let us first state the complexity gap theorem itself. We give here a formulation which is slightly different than, but essentially equivalent to the original one [48].

Theorem 3.5 *Complexity Gap*

We are given a second order existential ($SO\exists$) sentence ψ *of predicate logic that* fails in all finite models *(in the original formulation first order sentence is used, but the existential quantification over function or/and relation symbols is assumed implicitly). There is a procedure which translates the sequence of sentences* $A_n :=''\ \psi$ *has a model of size* n'' *into an unsatisfiable set* $C_{\psi,n}$ *of clauses. The sequence* $C_{\psi,n}$ *is uniformly generated (in the sense of [49])and its size is bound by a polynomial in* n*. The complexity gap theorem states that either 1 or 2 holds:*

1. *The sequence $C_{\psi,n}$ have polynomial size in n tree-resolution refutations.*
2. *There exists a positive constant a such that for every n each tree-resolution refutation of $C_{\psi,n}$ has to contain at least 2^{an} clauses.*

Furthermore, 2 holds if and only if ψ has an infinite model.

Thus the gap is between polynomial and exponential size proofs and shows that no super-polynomial (e.g. $2^{\theta(\log^p n)}$ for some $p > 1$) and sub-exponential (e.g. $2^{\theta(n^{\varepsilon})}$ for some $0 < \varepsilon < 1$) optimal proofs can appear.

Let us now consider the following encoding of (the negation of) PHP_n^{n+1} as a $SO\exists$ sentence (given also in [48])

$$\exists f\ \left(\left(\forall x, y\ (x = y) \vee (f(x) \neq f(y))\right) \wedge \left(\exists c \forall x\, f(x) \neq c\right)\right).$$

The complexity gap theorem gives only a $2^{\Omega(n)}$ lower bound, whereas we have shown that its real complexity is $2^{\theta(n \log n)}$. We have also shown that any, not necessary optimal, proof of PHP_n^{n+1} is of that size.

Another example we consider is *the minimum element principle*, saying that a total order always has a minimal element. Its negation can be encoded as

$$\exists L\ \left(\left(\forall x\, \neg L(x,x)\right) \wedge \left(\forall . x, y\ \left((x = y) \vee L(x,y) \vee L(y,x)\right)\right) \wedge\right.$$
$$\left.\left(\forall x, y, z\ \left(L(x,y) \wedge L(y,z)\right) \rightarrow L(x,z)\right) \wedge \left(\forall x \exists y\, L(y,x)\right)\right).$$

Here $L(x,y)$ stands for $x < y$. It can be easily shown (the lower bound also follows from theorem 3.5) that the optimal tree-resolution proof of the minimum element principle, ME_n (n is the number of elements), is $2^{\theta(n)}$. On the other hand, one can construct a proof which is as bad as possible, i.e. of size $2^{\theta\left(n^2\right)}$. There is also a *short, i.e. polynomial size, general resolution* proof of ME_n.

These two examples motivate the following two conjectures. The first one states that there is a second gap, while the second gives a characterisation of both gaps in terms of optimal and worst-case tree-resolution refutation. It also relates the gaps for tree- and general resolution.

Conjecture 3.6 *Given a $SO\exists$ sentence φ of predicate logic that* fails in all finite and infinite models*, and denote its translation (the same as in the theorem 3.5) into propositional logic by $C_{\psi,n}$. Then either 1, 2 or 3 holds:*

1. *The sequence $C_{\psi,n}$ have polynomial size in n tree-resolution refutations.*
2. *There is a refutation of $C_{\psi,n}$ of size 2^{an} for some positive constant a.*
3. *There is a positive constant b such that for every n each tree-resolution refutation of $C_{\psi,n}$ has to contain at least $2^{bn \log n}$ clauses.*

Conjecture 3.7 *Under the assumptions of the previous conjecture:*

In the second case $C_{\psi,n}$ has both a polynomial size general resolution proof and a worst-case tree-resolution proof, significantly worse than the optimal one, i.e. of size $2^{\Omega\left(n^2\right)}$.

In the the third case any general resolution proof of $C_{\psi,n}$ is of size $2^{\Omega(n)}$, and any tree-resolution proof is polynomially related to the optimal one.

Chapter 4

Resolution Width-Size trade-offs for the Pigeon-Hole Principle

We prove the following two results:

(1) There is a resolution proof of the Weak Pigeon-Hole Principle, $WPHP_n^m$ of size $2^{O\left(\frac{n\log n}{\log m}+\log m\right)}$ for any number of pigeons m and any number of holes n.

(2) Any resolution proof of $WPHP_n^m$ of width $\left(\frac{1}{16}-\varepsilon\right)n^2$ has to be of size $2^{\Omega(n)}$, independently from m.

These results give not only a resolution size-width tradeoff for the Weak Pigeon-Hole Principle, but also almost optimal such trade-off for resolution in general. The upper bound (1) may be of independent interest, as it has been known for the two extreme values of m, $m = n+1$ and $m = 2^{\sqrt{n\log n}}$, only.

4.1 Introduction

The Pigeon-Hole Principle is one of the simplest and, at the same time, the most important combinatorial principles. Its traditional formulation, denoted usually by PHP_n^m, states that there is no *injective* map from a finite m-element set into a finite n-element set if $m > n$. The Pigeon-Hole Principle also has the distinction to be the first and, since then, the most used combinatorial statement in propositional proof complexity. More specifically, it has been used quite a number of times in proving *lower bounds* for concrete *propositional proof systems.*

As the present paper is concerned with *resolution proofs* of PHP_n^m, we will briefly survey the history of proving resolution lower bounds for the Pigeon-Hole Principle. Everything started with the seminal Haken's result [31], that *any resolution proof* of PHP_n^{n+1} is of size $2^{\Omega(n)}$. The proof has been generalised and simplified in [16], [7], [11]. For quite a while, the best known result had been a $2^{\Omega\left(n^2/m\right)}$ lower bound from [16], thus having left the case $m = \Omega\left(n^2/\log n\right)$ as an important open problem in resolution proof complexity. Recently, a $2^{\Omega(n^{\varepsilon})}$ lower bound on *any regular-resolution proof* of PHP_n^m has appeared in [38]. Shortly after that, the problem has finally been solved by Raz in [42], and further strengthened and improved by Razborov in [43], [44].

All the mentioned proofs use, explicitly or implicitly, the relation between width (or "pseudo-width", as defined by Razborov) and size of the proof. This relation, used in almost all resolution lower bounds, not only for the Pigeon-Hole Principle, was extracted and stated precisely in [11]. On the other hand, nobody has studied the question whether there is a width-size trade-off in resolution proofs of some concrete statements. That is, can one pay for decreasing the size by increasing the width? Our paper answers the question as far as the Pigeon-Hole Principle is concerned.

We first prove an upper bound, i.e. construct a resolution proof of $WPHP_n^m$ of size $2^{O\left(\frac{n\log n}{\log m}+\log m\right)}$ for any m and n. Such an upper bounds have been known so far only for the two extreme cases $m = n+1$ and $m = 2^{\sqrt{n\log n}}$ (see [15]). Our result matches these, and, moreover, we believe that it is the exact resolution proof complexity of $WPHP_n^m$ for all m and n .

We also prove a $2^{\Omega(n)}$ lower bound on any resolution proof of the weak pigeon-hole principle, $WPHP_n^m$, when the width is bounded by $\left(\frac{1}{16}-\varepsilon\right)n^2$. Unlike the general lower bound in [43], it holds independently from the number of the pigeons m.

These two results not only give a resolution width-size trade-off for the Weak Pigeon-Hole Principle, but also have the following interesting consequence. Letting $m = 2^{\sqrt{n\log n}}$, we get a tautology on N variables which is provable in resolution within polylog (N) width. Any such proof however is of super-polynomial in N size. At the same time, there is a proof of poly (N) size and width N. This is asymptotically, i.e. modulo the degrees of the polynomials, the best resolution width-size trade-off, one could hope to prove.

The rest of the paper is organised as follows. In the section 4.2 we introduce the concepts, denotations and techniques. In the section 4.3 we show two trivial results, for the sake of the completeness only. The upper bound for $WPHP_n^m$ is shown in the section 4.4. The lower bound, when the width is restricted, is proven in the section 4.5 Finally, in the section 4.6, we discuss some open questions.

4.2 Preliminaries

We first introduce some denotations and give some definitions. $[n]$ denotes the set $\{1, 2, \dots n\}$, and $[n : m]$ denotes $\{n + 1, n + 2, \dots m\}$.

A *literal* is either a propositional variable or the negation of a propositional variable. A *clause* is a set of literals. It is satisfied by a truth assignment if at least one of its literals is true under this assignment. A set of clauses is *satisfiable* if there exists a truth assignment satisfying all the clauses.

PHP_n^m denotes the claim that there is no *injective map* from a set of size m to a set of size n, where $m > n$. We encode its negation as the following set of clauses

1. $\{p_{i,j} \mid j \in [n]\}$ for every pigeon $i \in [m]$

2. $\left\{\overline{p}_{i,k}, \overline{p}_{j,k}\right\}$ for every hole $k \in [n]$ and pigeons $i, j \in [m]$, $i \neq j$

Although we consider the *injective PHP,* all the results and proofs from the paper remain valid for the *functional* (where the map is required to be a function) and *bijective PHP*, too.

Resolution is a proof system designed to *refute* given set of clauses i.e. to prove that it is unsatisfiable. This is done by means of the resolution rule

$$\frac{C_1 \bigcup \{v\} \quad C_2 \bigcup \{\overline{v}\}}{C_1 \bigcup C_2}.$$

Thus, we can derive a new clause from two other clauses that contain a variable and its negation respectively. The goal is to derive the empty clause from the initial ones. Anywhere we say we *prove* some proposition, we mean that first we take its negation in a clausal form and then resolution is used to refute these clauses. Sometimes, for technical reasons only, we could also use the *weakening* rule

$$\frac{C_1}{C_1 \bigcup C_2}.$$

There is an obvious way to represent every resolution refutation as a directed acyclic graph whose nodes are labelled by clauses. The sources, i.e. the vertices with no incoming edges, are the initial clauses. The only sink, i.e. the vertex with no outgoing edges, is the empty clause. We then define *the size* of a proof to be the number of internal vertices, i.e. non-leaves, which is equal to the number of applications of the resolution rule. We do not however count the number of weakenings if any, as they are not essential and can be removed from the proof.

A very important technique, we use to prove lower bounds on proofs, is considering a proof as a *Prover-Adversary game.* It is first introduced in [41] and developed further in [40] especially for resolution.

There are two players, named *Prover* and *Adversary.* An unsatisfiable set of clauses is given. Adversary claims wrongly that there is a satisfying assignment. Prover's task is to convict him in lying. A *position* in the game is a partial assignment of the propositional variables. The game start from the empty position. Prover has two kind of moves:

1. She queries a variable, whose value is unknown in the current position. Adversary answers, and the position then is extended with the answer.

2. She forgets a value of a variable, which is known. The current position is then reduced, i.e., the variable value becomes unknown.

The game is over, when the current partial assignment falsifies one of the clauses. Prover then wins, having shown a contradiction.

The link to resolution proofs is easily seen. We take such a proof, reverse all the edges in its graph, and label all the vertices by the negation of the corresponding clauses. What we get is a description of Prover's *winning strategy.* Any node in the new graph corresponds to a position in the game. The negation of the corresponding clause is a conjunction of literals which defines the current partial assignment. The variable to be queried by Prover at this position is the variable resolved in the proof. The leaves in the new graph are negation of the initial clauses, i.e. arriving there means that an assignment falsifying such a clause is found, and therefore Prover wins the game.

A *deterministic* Adversary's strategy corresponds to a *single path* in the proof's graph. Therefore, he has to use a *randomised strategy* (called "super-strategy" in Pudlak's paper) in order to enforce a *big enough subgraph.* Any such Adversary's strategy can be used to prove a lower bound on any Prover's strategy description, and therefore on any resolution proof.

4.3 Two simple proofs of the Pigeon-Hole Principle.

In this section we construct two resolution proofs. They are stated and proven for the ordinary Pigeon-Hole Principle, PHP_n^{n+1} as one can always prove the Weak Pigeon-Hole Principle by restricting it to the first $n+1$ pigeons. The first proof shows that $WPHP_n^m$ can be proven in very small width which is also optimal. The second one is the optimal-size proof for the ordinary pigeon-hole principle. We will use it as a basis case in constructing a smaller proof in the case $m \gg n$.

Proposition 4.1 *There is a resolution proof of PHP_n^{n+1} of (optimal) width n and size $2^{O(n \log n)}$.*

Proof The proof contains n stages, numbered from n to 0. For each k, the k-th stage encodes the statement "there is no injective mapping of the first k pigeons into the holes". The corresponding clauses are

$$\left\{\overline{p}_{1,\pi(1)}, \overline{p}_{2,\pi(2)}, \ldots \overline{p}_{k,\pi(k)}\right\},$$

where π is any injective function from $[k]$ into $[n]$. Thus the stage 0 consists of the empty clause only. The n-th stage clauses can be derived by consecutively resolving the axiom $\{p_{n+1,1}, p_{n+1,2}, \ldots p_{n+1,n}\}$ with the axioms $\left\{\overline{p}_{n+1,\pi(1)}, \overline{p}_{1,\pi(1)}\right\}$, $\left\{\overline{p}_{n+1,\pi(2)}, \overline{p}_{2,\pi(2)}\right\}$,... and $\left\{\overline{p}_{n+1,\pi(n)}, \overline{p}_{n,\pi(n)}\right\}$ for every injective function $\pi : [n] \to [n]$.

What remains is to show how to go from the stage $k+1$ to the stage k. Given an injection $\pi : [k] \to [k]$, we shall derive the clause

$$\left\{\overline{p}_{1,\pi(1)}, \overline{p}_{2,\pi(2)}, \ldots \overline{p}_{k,\pi(k)}\right\} \tag{4.1}$$

from axioms and the $k+1$-th stage clauses

$$\left\{\overline{p}_{1,\pi(1)}, \overline{p}_{2,\pi(2)}, \ldots \overline{p}_{k,\pi(k)}, \overline{p}_{k+1,j}\right\}, \tag{4.2}$$

for all $j \in [n] \setminus \mathrm{Range}\,(\pi)$. The derivation is as follows. We first use the axioms $\{p_{k+1,1}, p_{k+1,2}, \ldots p_{k+1,n}\}$ and $\left\{\overline{p}_{k+1,\pi(1)}, \overline{p}_{1,\pi(1)}\right\}$, $\left\{\overline{p}_{k+1,\pi(2)}, \overline{p}_{2,\pi(2)}\right\}$, ... $\left\{\overline{p}_{k+1,\pi(k)}, \overline{p}_{k,\pi(k)}\right\}$ to obtain

$$\left\{\overline{p}_{1,\pi(1)}, \overline{p}_{2,\pi(2)}, \ldots \overline{p}_{k,\pi(k)}, p_{k+1,j_1}, p_{k+1,j_2}, p_{k+1,j_{n-k}}\right\},$$

where $\{j_1, j_2, \dots j_{n-k}\} = [n] \setminus \text{Range}(\pi)$. We then consecutively resolve it with the clauses (4.2) to "kill" the indices $j_1, j_2, \dots j_{n-k}$ and finally get the desired clause (4.1).

All the clauses used in the proof are of width less than or equal to n. The size can be estimated as follows. For every k, the k-th stage contains $\frac{n!}{(n-k)!}$ clauses, and in deriving each k-stage clause we have used n resolution steps. The overall number of resolution steps therefore is

$$n \sum_{k=0}^{n} \frac{n!}{(n-k)!} \sim enn! = 2^{O(n \log n)}.$$

□

Proposition 4.2 *There is a resolution proof of PHP_n^{n+1} of (optimal) size $2^{n+O(\log n)}$ and width $\frac{1}{4}n^2 + O(n)$*

Proof The proof contains n stages, the k-th one encoding the statement "there is no injective mapping of the first $k+1$ pigeons into any k-the element subset of $[n]$". The corresponding set of clauses is

$$P_{1,S} \bigcup P_{2,S} \bigcup \dots P_{k+1,S},$$

where S is any $n-k$-element subset of $[n]$, and $P_{i,S}$ is defined as $P_{i,S} := \{p_{i,j} \mid j \in S\}$. Thus the stage 0 consists of the single axiom $\{p_{1,1}, p_{1,2}, \dots p_{1,n}\}$, and the last stage, the n-th one, consists of the empty clause only.

What remains is to show how to go from the stage $k-1$ to the stage k. Given any $n-k$-element set $S \subseteq [n]$ we shall derive the clause

$$P_{1,S} \bigcup P_{2,S} \bigcup \dots P_{k+1,S} \tag{4.3}$$

from axioms and the $k-1$-th stage clauses

$$P_{1,S \bigcup \{j\}} \bigcup P_{2,S \bigcup \{j\}} \bigcup \dots P_{k,S \cup \{j\}}, \tag{4.4}$$

for all $j \in [n] \setminus S$. The derivation is as follows. For each $j \in [n] \setminus S$, we consecutively resolve the clause (4.4) with the clauses $\{\overline{p}_{1,j}, \overline{p}_{k+1,j}\}$, $\{\overline{p}_{2,j}, \overline{p}_{k+1,j}\}, \dots$ $\{\overline{p}_{k,j}, \overline{p}_{k+1,j}\}$ to obtain

$$P_{1,S} \bigcup P_{2,S} \bigcup \dots P_{k+1,S} \cup \overline{p}_{k+1,j}. \tag{4.5}$$

We then consecutively resolve these with the axiom $\{p_{k+1,1}, p_{k+1,2}, \dots p_{k+1,n}\}$ to get the desired clause (4.3).

We shall estimate the size and the width of the above resolution proof. The k-th stage consists of 2^{n-k} clauses, and in deriving each of these we have used $k^2 + k$ resolution steps. Thus the size of the proof is

$$\sum_{k=1}^{n} \left(k^2 + k\right) 2^{n-k} = O(n^2 2^n) = 2^{n+O(\log n)}.$$

The maximum width at the k-th stage is achieved when resolving the first clause of (4.5) with the axiom, and it is $(k+1)(n-k) + n - 2$. It is easy to see that the maximum over k is achieved near to $k = \frac{n-1}{2}$, and it is $\frac{1}{4}n^2 + O(n)$.□

4.4 The upper bound

In this section we construct the following resolution proof of the Weak Pigeon-Hole Principle.

Theorem 4.3 *There is a resolution proof of $WPHP_n^m$ of size $2^{O\left(\frac{n\log n}{\log m}+\log m\right)}$ for every m and n .*

Proof We construct the proof recursively as follows. We divide the pigeons into $\frac{m}{i}$ blocks of i pigeons in each block. We divide the holes into two groups of size j and $n-j$, respectively (the figure 4.1). We first take each block of i

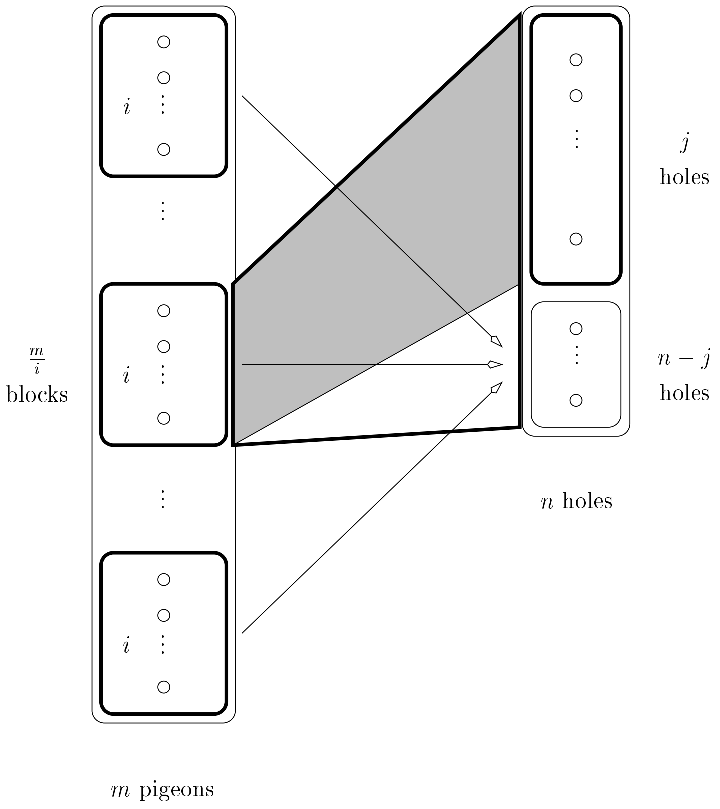


Figure 4.1: The recursion

pigeons against the first group of j holes, i.e. we consider PHP_j^i (the grey area on the figure 4.1), provided that $i > j$. As there are additional $n-j$ holes, we can derive a clause saying that at least one pigeon from the block goes to the second group of holes. We now consider each block as a single pigeon against the second group of holes (shown by arrows on the figure 4.1) , i.e. we have $PHP_{n-j}^{\frac{m}{i}}$, assuming $\frac{m}{i} > n-j$. Strictly speaking, this is not an instance of the Pigeon-Hole Principle. The propositional variables p_{ij} are now replaced by big disjunctions, and therefore the negations $\overline{p}_{ij}$ are now big conjunctions. This is a potential problem, because a resolution proof of the Pigeon-Hole principle can contain clauses having many negative literals. If we try to transform directly such a proof into a proof of the new version of PHP, where we have blocks instead of single pigeons, the expansion of these clauses would cause exponential blow-up. However, the difficulty can be overcame by considering only proofs that contains small (constant) number of negative literals in each clause. Indeed, the proof from proposition 4.2, we shall use as a basis case in our construction, is of such kind. By expanding the clause and simulating each resolution step, we multiply the size of the proof by only a polynomial (in this particular case quadratic) factor in the block-size. Therefore, denoting by $S(m,n)$ the size of the resolution proof of PHP_n^m, obtained in this way we get

$$S(m,n) \leq \frac{m}{i} S(i,j) + i^2 S\left(\frac{m}{i}, n-j\right).$$

We now let $i = \sqrt{m}$ and $i = \frac{n}{2}$and get

$$S(m,n) \leq \sqrt{m} S\left(\sqrt{m}, \frac{n}{2}\right) + mS\left(\sqrt{m}, \frac{n}{2}\right) \leq m^2 S\left(\sqrt{m}, \frac{n}{2}\right).$$

After k iterations of the above, we get

$$S(m,n) \leq m^{2\left(1+\frac{1}{2}+\ldots+\frac{1}{2^{k-1}}\right)} S\left(m^{\frac{1}{2^k}}, \frac{n}{2^k}\right) \leq m^4 S\left(m^{\frac{1}{2^k}}, \frac{n}{2^k}\right).$$

Let us now substitute $\log \frac{\log m}{\log n}$ for k:

$$S(m,n) \leq m^4 S\left(n, \frac{n \log n}{\log m}\right).$$

By Proposition 4.2, $S\left(n, \frac{n \log n}{\log m}\right)$, which is the basis case of the recursion, can be bounded by $2^{O\left(\frac{n \log n}{\log m}\right)}$. Therefore we have

$$S(m,n) \leq 2^{O\left(\frac{n \log n}{\log m} + \log m\right)},$$

as claimed. □

4.5 The lower bound

We shall prove the following

Theorem 4.4 *For any positive constant ε, every resolution proof of $WPHP_n^m$ of width $\left(\frac{1}{16} - \varepsilon\right) n^2$ has to be of size $2^{\Omega(n)}$*

Proof We first describe Adversary's strategy. He divides the set of holes into two equally big sets H_1 and H_2 of size $\frac{n}{2}$ each (this does not need to be at random), and then assign each pigeon to either H_1 or H_2, independently at random with probability $\frac{1}{2}$. At this stage, Adversary does not assign a particular hole to any pigeon, so we say that all the pigeons are *floating*. During the game, a pigeon becomes *fixed* iff either an "yes" answer or at least $\frac{n}{16}$ "no" answer about it are present in the current position. "Fixed" means that a hole is assigned to the pigeon. We can now explain Adversary replies to Prover's moves. There are several cases to be considered

1. Prover "forgets" some information. This is the easiest case. Adversary may need only to change the status of some pigeons from "fixed" to "floating".
2. Prover makes a query about a pigeon P. Assuming w.l.o.g. that P has been assigned to H_1, Adversary answers as follows:
   (a) P is fixed: consistently with the assigned hole.
   (b) The query is about a hole in H_2: "no".
   (c) P is floating and the query is about a hole in H_1: if the number of "no" answers, involving P and holes in H_1, in the current position, is less that $\frac{n}{4} - 1$, Adversary answers "no". Otherwise, he assigns the first empty hole in $H_1$to P, and then answers consistently with the assignment. P becomes fixed.

It is easy to see that Adversary can play until there are $\frac{n}{4}$ fixed pigeons in either part. Indeed, the only danger is when a pigeon P changes the status from "floating" to "fixed". Exactly $\frac{n}{4}$ holes are explicitly forbidden by "no" answers at this stage, while less than $\frac{n}{4}$ holes are implicitly forbidden, because they are assigned to the fixed pigeons. Thus there is an empty hole to be assigned to P.

Let us now consider the situation when Adversary gives up, i.e. there are exactly $\frac{n}{4}$ in one of the parts, either H_1 or H_2. There have to be at least $4\varepsilon n$ "yes" fixed pigeons, as otherwise there would be more that $\left(\frac{1}{4} - 4\varepsilon\right) n$ busy "no" pigeons and therefore the size of the position would be bigger than $\left(\frac{1}{16} - \varepsilon\right) n^2$. As the parts H_1 and H_2 have been assigned to the pigeons independently at random with probability $\frac{1}{2}$, it follows that the probability that the $4\varepsilon n$ "yes" busy pigeons are consistent with the initial assignment, is $\frac{1}{2^{4\varepsilon n}}$, and therefore the description of Prover's winning strategy has to be at least $2^{4\varepsilon n}$. $\square$

4.6 Conclusion

We have proven a resolution width-size trade-offs for the Weak Pigeon-Hole Principle. There are, however, several open questions left.

The bound on the width, $\left(\frac{1}{16} - \varepsilon\right) n^2$, is too strong. At least for the ordinary Pigeon-Hole Principle, one should be able to prove a stronger result. We conjecture the following sharp-threshold.

Conjecture 4.5 *For any positive constant ε, every resolution proof of PHP_n^{n+1} of width $\left(\frac{1}{4} - \varepsilon\right) n^2$ has to be of size $2^{\Omega(n \log n)}$*

It is in agreement with the upper bounds we have proven, the propositions 4.1 and 4.2. In a preliminary version of this work, we claimed the above conjecture as a proven result. Unfortunately, it seems that one cannot get this result with an easy adaptation of the current lower-bound techniques.

Another interesting open problem is to tighten the gap between the known upper and lower bounds on the resolution proofs of the weak pigeon hole principle, $WPHP_n^m$. The best known lower bound, $2^{\Omega\left(\frac{n}{(\log m)^2}\right)}$, appears in [45]. We conjecture that our upper bound is optimal.

Conjecture 4.6 *Any optimal proof of the* $WPHP_n^m$ *is of size* $2^{\Omega\left(\frac{n\log n}{\log m}\right)}$ *for* $n < m \leq 2^{\theta\left(\sqrt{n\log n}\right)}$.

Bibliography

[1] M. Ajtai. The complexity of the pigeonhole principle. *Combinatorica*, 14(4):417–433, 1994. A preleminary version appeared at the 29th FOCS in 1998.

[2] M. Alekhnovich. Mutilated chessboard is exponentially hard for resolution. Manuscript, 2000.

[3] A.Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.

[4] P. Beame, R. Impagliazzo, J. Krajicek, T. Pitassi, and P. Pudlak. Lower bounds on hilbert's nullstellensatz and propositional proofs. In *Proceedings of the 35th Annual Symposium on Foundations Of Computer Science*, pages 794–806. IEEE, November 1994.

[5] P. Beame, R. Impagliazzo, J. Krajicek, T. Pitassi, P. Pudlak, and A. Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the 24th Annual ACM Symposium on Theory Of Computing*, pages 200–220. ACM, May 1992.

[6] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability of random k-cnf formulas. In *Proceedings of the 30th Annual ACM Symposium on Theory Of Computing*, pages 561–571. ACM, May 1998.

[7] P. Beame and T. Pitassi. Simplified and improved resolution lower bounds. In *Proceedings of the 37th annual IEEE symposium on Foundation Of Computer Science*, pages 274–282, 1996.

[8] P. Beame and T. Pitassi. Propositional proof complexity: past, present and future. Technical Report 67, ECCC, 1998.

[9] P. Beame and S. Riis. More on the relative strength of counting principles. In P. Beame and S. Buss, editors, *Proof Complexity and Feasible Arithmetics*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 13–35. 1998.

[10] E. Ben-Sasson. Hard examples for bounded depth frege. In *Proceedings of the 34st Annual ACM Symposium on Theory Of Computing*, May 2002. To appear.

[11] E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2), March 2001. A preleminary version appears at the 31st STOC, 1999.

[12] S. Buss. Polynomial size proofs of the pigeon-hole principle. *Journal of Symbolic Logic*, 57, 1987.

[13] S. Buss. *Handbook of Proof Theory*, chapter An Introduction to Proof Theory. Elsevier, 1998.

[14] S. Buss, D. Grigoriev, R. Impagliazzo, and T. Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. In *Proceedings of the 31st Annual ACM Symposium on Theory Of Computing*, pages 547–556. ACM, 1999.

[15] S. Buss and T. Pitassi. Resolution and the weak pigeonhole principle. In *Computer science logic (Aarhus, 1997)*, pages 149–156. Springer, 1998.

[16] S. Buss and G. Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62:311–317, 1988.

[17] V. Chvátal and E. Szemereédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, 1988.

[18] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory Of Computing*, pages 174–183. ACM, May 1996.

[19] S. Cook. The complexity of theorem proving procedures. In *Proceedingsof the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

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

[21] W. Cook, R. Coullard, and G. Turan. On the complexity of cutting plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.

[22] S. Dantchev. Width-size trade-offs for the pigeon-hole principle. In *The 17th annual IEEE Conference on Computational Complexity*. IEEE, May 2002.

[23] S. Dantchev and S. Riis. "planar" tautologies, hard for resolution. In *In proceedings of the 42nd annual symposium on Foundations Of Computer Science*. IEEE, October 2001.

[24] S. Dantchev and S. Riis. Tree resolution proofs of the weak pigeon-hole principle. In *In proceedings of the 16th annual IEEE Conference on Comutational Complexity*. IEEE, June 2001.

[25] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, (5):394–397, 1962.

[26] M. Davis and H. Putnam. A computing procedure for quantification theory. *Communications of the ACM*, (7):201–215, 1960.

[27] E. Friedgut. Sharp thresholds of graph proprties, and the k-sat problem. *Journal of the AMS*, 12(4):1017–1054, 1999.

[28] X. Fu and A. Urquhart. Simplified lower bounds for propositional proofs. *Notre Dame Journal of Formal Logic*, 37(4):523–544, 1996.

[29] A. Goerdt. A threshold for unsatisfiability. *Jornal of Computer and System Sciences*, 53:469–486, 1996.

[30] R. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the AMS*, (64):275–278, 1958.

[31] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[32] K. Iwama and S. Miyazaki. Tree-like resolution is superpolynomially slower than DAG-like resolution for the pigeonhole principle. In *Algorithms and computation (Chennai, 1999)*, Lecture Notes in Computer Science, pages 133–142. 1999.

[33] J. Krajíĉek. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, 1995.

[34] L. Levin. Universalnie zadachi perebora. *Problemi peredachi informacii*, IX(3):115–116, 1973. In Russian; translation as: Universal sequential search problems, Problems of Information Transmission 9,3, 265–266.

[35] L. Lovasz and A. Schrijver. Cones of matrices and set-functions and $0-1$ optimization. *SIAM Journal on Optimization*, 7(1):166–190, 1991.

[36] J. McCarthy. A tough nut for proof procdures. Stanford Artifical Intelligence Project Memo 16, July 1964.

[37] T. Pitassi. Propositional proof complexity and unsolvability of polynomial equations. In *Proceedings of International Congress of Mathematicians*, Berlin, 1998.

[38] T. Pitassi and R. Raz. Regular resolution lower bounds for the weak pigeonhole principle. In *Proceedings of the 33rd annual ACM Symposium on Theory Of Computing*, pages 347–355, 2001.

[39] P. Pudlak. On the complexity of propositional calculus. In *Sets and Proofs*, pages 197–218. Cambridge University Press, 1999. Invited papers from Logic Colloquium'97.

[40] P. Pudlák. Proofs as games. *American Mathematical Monthly*, pages 541–550, June-July 2000.

[41] P. Pudlák and S. Buss. How to lie without being (easily) convicted and the lengths of proofs in propositional calculus. In *Computer Science Logic'94*, volume 993 of *Lecture Notes in Computer Science*, pages 151–162, 1995.

[42] R. Raz. Resolution lower bounds for the weak pigeonhole principle. Technical Report 21, Electronic Colloquium on Computational Complexity, 2001. Avaliable at http://www.eccc.uni-trier.de/eccc/.

[43] A. Razborov. Improved resolution lower bounds for the weak pigeonhole principle. Technical Report 55, Electronic Colloquium on Computational Complexity, 2001. Avaliable at http://www.eccc.uni-trier.de/eccc/.

[44] A. Razborov. Resolution lower bounds for the weak functional pigeonhole principle. Technical Report 75, Electronic Colloquium on Computational Complexity, 2001. Avaliable at http://www.eccc.uni-trier.de/eccc/.

[45] A. Razborov. Resolution lower bounds for perfect matching principles. In *Proceedings of the 17th annual IEEE Conference on Comutational Complexity*. IEEE, May 2002. Avaliable at http://www.mi.ras.ru/ razborov/.

[46] S. Riis. *count*(q) does not imply *count*(p). *Annals of Pure and Applied Logic*, 90:1–56, 1997.

[47] S. Riis. *count*(q) versus the pigeon-hole principle. *Archive for Mathematical Logic*, 36(3):157–188, 1997.

[48] S. Riis. A complexity gap for tree-resolution. *Computational Complexity*, to appear.

[49] S. Riis and M. Sitharam. Generating hard tautologies using predicate logic and the symmetric group. *Logic Journal of the IGPL*, 8(6):787–795, 2000.

[50] M. Sipser. The history and status of the p versus np question. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 603–618. ACM, 1992.

[51] G. Tseitin. On the complexity of derivation in the propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 1968.

[52] A. Urquhart. Hard examples for resolution. *Jornal of the ACM*, 34(1):209–219, 1987.

[53] A. Urquhart. Resolution proofs of matching principles. *Annals of Mathematics and Artificial Intelligence*, 1998.