On the Combinatorics of SAT and the Complexity of Planar Problems

Navid Talebanfard

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

On the Combinatorics of SAT and the Complexity of Planar Problems

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

by
Navid Talebanfard
August 31, 2014

Abstract

In this thesis we study several problems arising in Boolean satisfiability ranging from lower bounds for SAT algorithms and proof systems to extremal properties of formulas. The first problem is about construction of hard instances for k-SAT algorithms. For PPSZ algorithm [40] we give the first construction of k-CNF instances that require running time at least $2^{(1-\epsilon_k)n}$, where $\epsilon_k = O(\log^2 k/k)$. For DPLL ([22], [21]) we construct unsatisfiable hard instances requiring a running time of the same form, except with $\epsilon_k = \tilde{O}(k^{-1/3})$. This improves on a previous construction which gives $\epsilon_k = \tilde{O}(k^{-1/4})$.

The depth of a resolution refutation is defined as the length of a longest path from the conclusion to an axiom. We relate variable space and depth in resolution and answering a question of Urquhart [53], we show that if a formula has a resolution refutation with variable space s, it also has a resolution refutation of depth $O(s^2 \log n)$. We also construct unsatisfiable formulas which are refutable in constant space but require logarithmic depth. This shows that at least for constant variable space the upper bound is sharp.

Another problem we consider is to give sharp bounds for the number of prime implicants of k-CNF formulas. For $k = 2$ we give a characterization of prime implicants and show that the maximum number of them is between $3^{n/3}$ and $(1+o(1))3^{n/3}$. For $k \geq 3$, we extend the Satisfiability Coding Lemma [41] and show that if each variable occurs in a constant number of clauses the number of prime implicant is bounded by $3^{(1-\Omega(1/k))n}$. We also show that there are k-CNF formulas that have at least $3^{(1-O(\log k/k))n}$ prime implicants.

In the final part of the thesis we study properties of graphs with bounded planar cutwidth. Using algebraic characterizations of bounded depth circuit classes we can show that the 2-coloring problem for such graphs is in $\mathsf{AC}^0[2]$ and the matching problem is in ACC^0.

Resumé

I denne afhandling studerer vi adskillige problemer angående Boolesk satisfiability, rangerende fra nedre grænser for SAT algoritmer og bevissystemer til extremale egenskaber af Booleske formler. Det første problem vi ser på omhandler konstruktion af svære instanser for k-SAT algoritmer. For PPSZ algoritmen [40] giver vi den første konstruktion af instanser der kræver kørselstid $2^{(1-\epsilon_k)n}$, hvor $\epsilon_k = O(\log^2 k/k)$. For DPLL algoritmen ([22], [21]) konstruerer vi uopfyldelige instanser der kræver en kørselstid af samme form, blot med $\epsilon_k = \tilde{O}(k^{-1/3})$.

Dybden af et resolutionsbevis er defineret til at være længden af den længste sti fra konklusionen til et af aksiomerne. Vi relaterer variabelplads og dybde for resolutionsbeviser og besvarer dermed et spørgsmål af Urquhart [53]: Vi viser at hvis en formel har et resolutionsbevis med variabelplads s, har den også et resolutionsbevis af dybde $O(s^2 \log n)$. Vi konstruerer også formler med resolutionsbeviser der bruger konstant plads, men som kræver resolutionsbeviser af logaritmisk dybde. Disse formler viser at for konstant variabelplads er vores øvre grænse tæt.

Vi forsøger endvidere at give tætte grænser for antallet af primimplikanter af k-CNF formler. For $k = 2$ giver vi en karakterisation af primimplikanterne, og viser at deres maksimale antal er mellem $3^{n/3}$ og $(1 + o(1))3^{n/3}$. For $k \geq 3$ generaliserer vi Satisfiability Coding lemmaet og viser at hvis hver variabel optræder i et konstant antal klausuler, er antallet af primimplikanter begrænset af $3^{(1-\Omega(1/k))n}$. Vi viser ogsåat der findes k-CNF formler der har mindst $3^{(1-O(\log k/k))n}$ primimplikanter.

I den sidste del af afhandlingen studerer vi grafer med begrænset planær snitbredde. Ved brug af kendte algebraiske karakterisationer af kompleksitetsklasser for netværk af konstant dybde, viser vi at tofarvningsproblemet for sådanne grafer er i kompleksitetsklassen $\mathsf{AC}^0[2]$ og at matchingproblemet er i kompleksitetsklassen ACC^0.

Acknowledgments

I would like to thank my advisors, Kristoffer Arnsfelt Hansen and Peter Bro Miltersen for their continuous support throughout my studies. I would also like to thank my friends and colleagues at the Center for the Theory of Interactive Computation both at Aarhus University and Tsinghua University for making my time in Aarhus and Beijing memorable: Simina Brânzei, Joshua Brody, Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, Pavel Hubáček, Kevin Matulef, Periklis Papakonstantinou, Dominik Scheder, John Steinberger, Elias Tsigaridas, and Iddo Tzameret.

I was fortunate to visit the Theory group at KTH Royal Institute of Technology and benefited from conversations with Per Austrin, Massimo Lauria, Mladen Mikša, Jakob Nordström, and Marc Vinyals, whom I would like to sincerely thank.

I am also grateful to my co-authors Ilario Bonacina, Shiteng Chen, Kristoffer Arnsfelt Hansen, Balagopal Komarath, Jayalal Sarma, Dominik Scheder, Sven Skyum and Bangsheng Tang.

Navid Talebanfard,
Aarhus, August 31, 2014.

Contents

Abstract **i**

Resumé **iii**

Acknowledgments **v**

Contents **vii**

1 Introduction **1**

2 Strong Exponential Lower Bounds for k-SAT Algorithms **7**
2.1 Lower Bounds for DPLL 7
2.2 Lower Bounds for PPSZ 13

3 Variable Space versus Depth in Resolution **31**
3.1 introduction 31
3.2 Main Result 32

4 On the Structure and the Number of Prime Implicants of k-CNF Formulas **35**
4.1 $k = 2$ 36
4.2 $k \geq 3$ 39

5 Circuit Complexity of Properties of Graphs with Bounded Planar Cut-width **43**
5.1 Introduction 43
5.2 Preliminaries 47
5.3 Upper bounds 48
5.4 Hardness Results 60

6 Open Problems **67**

Bibliography **69**

Chapter 1

Introduction

Boolean satisfiability and constant depth computation have been major topics of study in computational complexity and shown to be intimately related. Boolean satisfiability stands at the heart of computational complexity theory, not only for its NP-completeness, perhaps rather for its impact on shaping essentially every branch in theoretical computer science. The magic probably lies in how simply it can be stated: we have a function represented by a formula, a circuit or any structure that you like, decide whether there is an input to this function that makes it evaluate to one. The easiest way to check this is to go through all possible inputs and hence costing an exponential number of steps. Williams brought constant depth circuits to the realm of satisfiability in [57] when he showed that beating this bound by even a small polynomial factor has enormous consequences, namely, one can get lower bounds for properly defined circuit classes. In [58] he in fact succeeded in using this approach by obtaining an improved satisfiability algorithm for ACC^0 circuits and hence showing that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$.

In the other direction, that is bringing satisfiability algorithms in when studying bounded depth circuits, one should not forget the work of Paturi, Pudlák and Zane [41], who proved sharp depth-3 circuit lower bounds and from their lower bound technique, they obtained the currently best known k-SAT algorithm. The significance of depth-3 circuits was already known to Valiant in 1979. Even though we know that almost all functions require exponentially large circuits, the best circuit lower bounds for explicit functions are only linear. Valiant [54] showed that a linear size boolean circuit of logarithmic depth could be written as a depth-3 circuit of subexponential size where the bottom fanin is n^ϵ for any $\epsilon > 0$ (see [56] for a nice account of this). He gets a bottom fanin of $O(1)$ when the original circuit is series-parallel. One can think of such circuits as disjunctions of n^ϵ-CNFs or k-CNFs for a constant k. Therefore to prove explicit superlinear circuit lower bounds of logarithmic depth, one could go about showing that an explicit function respresented as a disjunction of bounded width CNFs requires exponentially many disjuncts.

One might hope that understanding k-CNFs could eventually help us obtain circuit lower bounds through Valiant's reduction, and this is exactly what was done in [41].

The Satisfiability Coding Lemma [41] gives a bound on the number of *isolated* satisfying assignments of a k-CNF, where isolated means that if we flip the value of any single variable, the formula is not satisfied anymore. This lemma was used to prove sharp circuit lower bounds for parity, and surprisingly it gave rise to a randomized k-SAT algorithm running in time $2^{(1-\Omega(1/k))n}$. There is much evidence that this is not a coincidence; several improved satisfiability algorithms were recently obtained from circuit lower bound techniques. Impagliazzo, Matthews and Paturi [32] extended the Switching Lemma and gave a satisfiability algorithm for linear size AC^0 circuits. Santhanam [48] adopted an old lower bound argument to give a satisfiability algorithm for linear size De Morgan formulas.

Let us get back to depth-3 circuits and the complexity of k-SAT. Even though there are no formal connections between the two, it is tempting to envision an argument that translates a depth-3 ciruit lower bound proof into a k-SAT algorithm, possibly through a much stronger Satisfiability Coding Lemma. Falling short of this however, we can ask what the true complexity of k-SAT is. Impagliazzo, Paturi and Zane [34] formulated the so-called Exponential Time Hypothesis (**ETH**) which states that there is no subexponential time algorithm for k-SAT. Impagliazzo and Paturi [33] stated a stronger form of this hypothesis (Strong **ETH** or **SETH**) which says that k-SAT in fact gets harder and harder as k grows. More formally let σ_k be the infimum of all ϵ such that there exists k-SAT algorithm running in time $2^{(1-\epsilon)n}$. Then **SETH** states that σ_k tends to 0 as k grows. Assuming **ETH** they showed that σ_k is increasing infinitely often. It is an open problem whether **ETH** implies **SETH**.

This thesis is a two-fold attempt at understanding the nature of satisfiability and constant depth computation. Below we give an overview of this work:

Boolean Satisfiability

Proving **ETH** or **SETH** is much stronger than separating P from NP, and hence at the moment there is no hope of proving either of the two. However, one can ask whether known satisfiability algorithms have running times consistent with these hypotheses, that is if we can construct hard instances for specific algorithms making them run for exponentially many steps.

We consider two types of algorithms: branching algorithms or DPLL ([22], [21]) which are backtracking algorithms that successively assign values to variables and proceed, and encoding algorithms such as PPZ and PPSZ [40]. It is usually not too difficult to obtain exponential lower bounds consistent with **ETH** using connection with proof complexity. Resolution is a sound and

complete proof system for refuting unsatisfiable CNF formulas using the following rule: from clauses $C \vee x$ and $D \vee \neg x$ we can infer $C \vee D$. A run of a DPLL algorithm on an unsatisfiable formula generates a tree-like resolution refutation of the formula and exponential size lower bounds have been known for resolution in proof complexity for a long time (see e.g. [28], [52]). Here tree-like resolution refers to a restriction of resolution in which we cannot use a clause more than once, and if we need to do it, we should use a fresh copy of the clause. Obtaining strong exponential lower bounds consistent with **SETH** has been a great challenge even for tree-like resolution which was only settled by Pudlák and Impagliazzo [43], some fifteen years after the first exponential loweer bounds were obtained for resolution. A now standard technique due to Ben-Sasson and Wigderson [12] to prove resolution size lower bound is to prove width lower bounds, that is showing that in any refutation of a formula, a large clause has to appear. In particular they showed that logarithm of the size of a tree-like resolution refutation of a formula is at least the largest width needed to refute the same formula. But the best known width lower bounds have only been linear in the number of variables and not respecting k. In fact Ben-Sasson and Impagliazzo [11] showed that any formula which encodes an unsatisfiable linear system over $\mathbb{F}_2$, has a resolution refutation of width at most $n/2 + o(n)$ where n is the number of variables. Therefore one cannot hope to get strong exponential lower bounds from linear systems over $\mathbb{F}_2$ using just the width-size relationship. This was the main challenge in [44] which was circumvented. Only recently Beck and Impagliazzo [10] simplified this argument and gave formulas which require resolution width $(1 - o_k(1))n$ and hence strong exponential lower bounds for tree-like resolution. One of the main results of this thesis is a further simplification and improvement of this result. We modify of the construction of [10] and improve the $o_k(1)$ term.

Another parameter of DPLL algorithms is the depth of the corresponding branching tree. Urquhart [53] initiated a study of this parameter and showed that it is related to the resolution width. He also showed that it is always at least the number of variables needed to keep in memory in order to be able to carry the resolution proof on. Answering an open question of Urquhart we show that for formulas that have small variable space resolution refutations, there also exist small depth refutations.

In a quest to develop stronger encoding arguments for solutions of k-CNFs, we study the problem of bounding the number of prime implicants of k-CNF formulas. A *prime implicant* of a Boolean formula is a partial assignment of the variables that makes the function true, and if we unspecify any variable, the function is not constant anymore. We generalize the Satisfiability Coding Lemma to bound the number of prime implicants of k-CNF formulas in which every variable appears in a bounded number of clauses.

Constant Depth Computation

In this part we deviate from satisfiability a bit and study constant depth through an algebraic/geometric lens. Barrington's theorem [4] characterizes NC^1 as the class of functions computed by bounded-width branching programs of polynomial length. Shortly after this theorem was proved, an extension was given in [7] that characterized circuit classes NC^1, AC^0 and ACC^0 in terms of monoid programs. Barrington and Straubing [6] used these characterizations to prove a superlinear ACC^0 formula size lower bound for an explicit function. What is notable about this work is that everything is built on an elegant Ramsey theoretic argument showing that no read-once program over an aperiodic monoid computes parity.

There is a natural way to transform a monoid program to a branching program and vice versa, and hence one might wonder whether it is possible to get fine characterizations of AC^0 and ACC^0 in terms of branching programs. This in fact turns out to be the case if we put geometric restrictions on the branching programs. Barrington et al. [5] characterized AC^0 as the class of functions computed by upward planar branching programs of polynomial size, and if we have only planarity we get ACC^0 as proved by Hansen and Koucký [29]. It is diffcult not to envision lower bound proofs that employ these characterizations. Does planarity give any intuition why for instance parity is hard for AC^0, some intuition that the Switching Lemma cannot exploit? It is not clear at the moment how one can make use of planarity to prove lower bounds, but given that we have a characterization of ACC^0, a technique for proving AC^0 lower bounds assuming only upward planarity can potentially give rise to ACC^0 lower bound (note that the best ACC^0 lower bound is for an explicit function in NEXP), and thus we do not expect this approach to be an easy one, and yet unexplored.

We do however obtain non-trivial circuit upper bounds following this approach. We consider the graph of a bounded width planar branching program and ask if this graph satisfies certain properties. In [30] we study 2-colorability and the perfect matching problems, which are NC^1-complete for bounded width graphs without the planarity condition. But if we add planarity we can put 2-colorability in $\mathsf{AC}^0[2]$ and the perfect matching problem in ACC^0. We do not claim that such results are immediately related to circuit lower bounds, but we strongly believe that it is worthwhile experimenting with the range of techniques one can use in the geometric setting. Here is a rough outline of our proofs. Instead of constructing circuits directly we give monoid programs for each of these problems and use Barrington-Thérien Theorem. To show that the perfect matching problem is in ACC^0 we need to show that our monoid is solvable. We show that this monoid can only contain groups of odd order and hence by the Feit-Thompson theorem the result follows. In one of the steps of the proof of the fact that there are no groups of even order, we need to get a contradiction to the planarity assumption, and we do this via a

very intuitive use of Jordan's curve theorem.

Chapter 2

Strong Exponential Lower Bounds for k-SAT Algorithms

Since we do not yet know how to separate P from NP, we cannot prove unconditional super-polynomial lower bounds for satisfiability algorithms. However, we can try to do so for specific algorithms with the hope of getting insight into the nature of satisfiability. In this chapter we focus on lower bounds for the running time of certain SAT algorithms. We will discuss how one can use the structure of an algorithm to construct hard instances that make it run for many steps.

2.1 Lower Bounds for DPLL[1]

The DPLL algorithm [22] is a recursive algorithm that on a satisfiable formula F outputs a satisfying assignment and otherwise it rejects the input. At each step it chooses a variable x and simplifies $F^{[x\to 0]}$ and $F^{[x\to 1]}$ and then recurses on the obtained formulas. There are different variants of the DPLL algorithm which are determined by the rules choosing the variable at each stage and the simplification method.

Lower bounds for DPLL are usually obtained through its connection with proof complexity. Assume that the input F to a DPLL algorithm is unsatisfiable. Then one can think of a run of DPLL on F as a refutation of F, and in fact this refutation has a very simple structure, i.e., it is a resolution refutation of the formula.

Resolution (RES) is one of the most fundamental and extensively studied proof systems. Using this proof system one can refute unsatisfiable CNF formulas using the following inference rule

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

[1]This section is based on [15].

where C and D are disjunctions of literals and x is some variable. Every resolution refutation induces a DAG in the following way. There is a node for each clause appearing in the proof and every such node will be connected by an edge to the nodes corresponding to the two clauses from which this clause was derived. Given a formula φ we denote a RES refutation of φ using the notation $\varphi \vdash_\pi \bot$.

Tree-like resolution (treeRES) is referred to a subclass of resolution for which the induced DAG is in fact a tree. Let C be a clause, the *width of* C, $|C|$, is the number of literals appearing in C. The resolution *width* of a formula φ denoted by $\text{width}(\varphi \vdash \bot)$ is the minimum of the width of the largest clause appearing in any resolution refutation of φ, more formally

$$\text{width}(\varphi \vdash \bot) := \min_{\pi} \max\{|C| \ : \ \varphi \vdash_\pi \bot \wedge \text{C appears in } \pi\}.$$

The first strongly exponential lower bound for tree-like resolution was obtained by Pudlák and Impagliazzo.

Theorem 2.1.1 ([43]). *For every $k > 0$ there exists $\epsilon_k = O(1/k^{1/8})$ and a sequence of unsatisfiable k-CNFs that do not have tree-like resolution refutations of length less than $2^{(1-\epsilon_k)n}$ where n is the number of variables.*

It is not hard to see that any DPLL algorithm on an unsatisfiable formula F produces a tree-like resolution refutation of F. Thus the running time of the algorithm is lower bounded by the length of the shortest refutation of the input. Theorem 2.1.1 immediately implies a lower bound for DPLL.

Corollary 2.1.2 ([43]). *For every k there exists $\epsilon_k = O(1/k^{1/8})$ and a sequence of unsatisfiable k-CNF formulas on which any DPLL algorithm runs for at least $2^{(1-\epsilon_k)n}$ steps where n is the number of variables.*

A recent construction by Beck and Impagliazzo [10] improves this to $\epsilon_k = \widetilde{O}(1/k^{1/4})$, where the $\widetilde{O}$ notation is hiding log factors.

In this section we clarify and simplify the result in [10] making a further improvement to $\epsilon_k = \widetilde{O}(1/k^{1/3})$. One can also ask how far this improvement can go. Using the Switching Lemma, we can show that for every unsatisfiable k-CNF on n variables, there exists a tree-like resolution of size at most $2^{\left(1-\Omega(1/k)\right)n}$. A similar argument was used by Miltersen et al. [37] to prove upper bounds on the size of decision trees for k-CNFs. However, we are not aware of any adaptation of this in the proof complexity literature and will hence present a formal account of this observation.

Relationship with [10] As already said this paper simplifies, clarifies and improves the results from Beck and Impagliazzo [10]. They construct unsatisfiable linear systems of equations over $\mathbb{F}_p$ with a certain kind of expansion property. Then they encode each variable from $\mathbb{F}_p$ using a sum of roughly p^2

boolean variables and show that with this encoding the linear system requires very large resolution width.

The key property of this representation used in the proof is the following: let $z = \sum_{i=0}^{p^2} x_i$, where x_i are boolean variables. Even setting a lot of variables (i.e. $p^2 - p$) and still we can obtain all possible $\mathbb{F}_p$ values for z setting the remaining variables.

In other words what Beck and Impagliazzo really require is just a disperser for a bit-fixing source that can extract $\log p$ bits even after many bits in the seed are fixed. Our contribution is thus to show that a random function satisfies this property (Lemma 2.1.7), and we use this function instead of the sum of p^2 boolean variables used by Beck and Impagliazzo. The arguments of [10] still goes through. Beck and Impagliazzo use roughly p^2 bits for each $\mathbb{F}_p$ variables, whereas with our construction we only require around p bits and hence we get the improvement.

Notations A *restriction* on a set of variables X is a mapping $\rho : X \to \{0, 1, \star\}$. We call a variable *unfixed by* ρ if it is assigned $\star$, and we call it *fixed* otherwise. The domain of ρ, denoted by $\operatorname{dom} \rho$, is the set of variables fixed by ρ and $|\rho| := |\operatorname{dom}(\rho)|$. For a function f, we define $f|_\rho$ to be the function after setting values to the fixed variables according to ρ. A *random restriction* leaving ℓ variables free can be obtained as follows: first pick a subset S of the variables of size $|X| - \ell$ uniformly at random, then set each $x \in S$ to either 0 or 1 with equal probability.

Size upper bound

A *decision tree for an unsatisfiable* k-CNF φ is a just a standard decision tree, except that each leaf is labeled with some clause C of φ with the condition that the unique (partial) assignment reaching that leaf falsifies C. Following Beame [8] we define the canonical decision tree. Given $\varphi = \bigwedge_i C_i$ consider an ordering of the variables and an ordering of the clauses. The *canonical decision tree* of φ, denoted by $T(\varphi)$, is inductively defined as follows: look at the first clause C of φ according to the ordering and assume $\varphi = C \wedge \varphi'$. Then do a full decision tree on the variables of C respecting the order of the variables. To the leaf corresponding to the restriction which falsifies C, we assign C. For other leaves corresponding to restriction σ, we replace the leaf with $T(\varphi'|_\sigma)$. Note that for an unsatisfiable φ, $T(\varphi)$ corresponds to a tree-like resolution refutation of φ. Notice that canonical decision trees in [8] are defined for general CNFs. Our main observation, here, is that we can adopt them for the unsatisfiable setting. We need the following variant of the Switching Lemma due to Beame [8].

Lemma 2.1.3 (Switching Lemma)**.** *Let φ be a k-CNF on n variables. Let ρ be a random restriction leaving ℓ variables free. The probability that the*

canonical decision tree of $\varphi|_\rho$ has depth bigger than d is at most $\left(\frac{7k\ell}{n}\right)^d$.

Theorem 2.1.4. *For any unsatisfiable k-CNF φ on n variables*

$$\text{size}_{\text{treeRES}}(\varphi \vdash \bot) \leqslant 2^{\left(1-\Omega(\frac{1}{k})\right)n}.$$

Proof. We first note that a tree-like resolution refutation can be thought of as a decision tree in the following sense (see e.g. [13]).

Then we follow an argument due to Miltersen et al. [37] who showed that every k-CNF has a decision tree representation of size $2^{\left(1-\Omega(\frac{1}{k})\right)n}$ and adjust it to the unsatisfiable setting. We set $\ell = n/14k$. By the Switching Lemma, for a $1 - 2^{-d}$ fraction of restrictions σ with $|\sigma| = n - \ell$, we know that the depth of $T(\varphi|_\sigma)$ is at most d.

Then, by an averaging argument, there exists a set $S \subseteq X$ with $|S| = n-\ell$ such that the same statement holds for all restrictions fixing variables only in S. We can construct a decision tree for φ as follows: first we do a full decision tree on variables in S; then for each leaf with the corresponding restriction σ, we append $T(\varphi|_\sigma)$ to that leaf. The number of leaves of this tree is upper bounded by

$$2^d 2^{n-\ell} + 2^{n-\ell-d}2^\ell.$$

Hence setting $d := \ell/2$ the number of leaves of the tree is upper bounded by $2^{n-\frac{\ell}{2}+1} = 2^{\left(1-\Omega(\frac{1}{k})\right)n}$. Hence we have the required upper bound on the size of the decision tree constructed. □

Size lower bound

Let $v = (v_1, v_2, \ldots)$ be a vector over $\mathbb{F}_p$, then with $\text{supp}(v)$ we denote the indices of v with non-zero entries, that is $\text{supp}(v) := \{i \ : \ v_i \not\equiv_p 0\}$.

In what follows we construct a system of linear equations over $\mathbb{F}_p$. Let m be the total number of equations, n the total number of variables, and $\{z_1, \ldots, z_n\}$ the variables taking values over $\mathbb{F}_p$. We use the letter E, with subscripts, to denote linear equations mod p, that is an expression of the form $\sum_j a_j z_j \equiv_p b$, with $a_j, b \in \mathbb{F}_p$. We denote with $\text{supp}(E)$ the set of non-zero a_js. The following definition is essentially the same from [10].

Definition 2.1.5 ((α, β, γ)-expander)**.** *Let $\alpha, \beta, \gamma \in \mathbb{R}_{\geqslant 0}$, $m, n \in \mathbb{N}$ and $\mathscr{E} := \{E_1, \ldots, E_m\}$, a set of m linear equations over $\mathbb{F}_p$. We say that $\mathscr{E}$ is an* (α, β, γ)-expander *if and only if*

$$\forall v \in \mathbb{F}_p^m, \ |\text{supp}(v)| \in [\alpha, \beta] \Rightarrow \left|\text{supp}\left(\sum_{i=1}^m v_i E_i\right)\right| \geqslant \gamma.$$

The next Proposition is just a particular case of Lemma 4.2 from [10].

Proposition 2.1.6. *There exists a set $\mathscr{E} := \{E_1, \ldots, E_m\}$ of linear equations in n variables over $\mathbb{F}_p$ such that:*

1. *$\mathscr{E}$ is unsatisfiable,*
2. *for each $E_i \in \mathscr{E}$ $|\operatorname{supp}(E_i)| \leqslant p^2$,*
3. *$\mathscr{E}$ is $(\delta n, 3\delta n, (1 - c\theta)n)$-expander, where $\delta = O(1/p)$, $\theta = \widetilde{O}(1/p)$ and c is a constant,*
4. *no subset of at most $3\delta n$ equations from $\mathscr{E}$ is unsatisfiable.*

Lemma 2.1.7. *Let θ be the parameter coming from Proposition 2.1.6. Then there exists a function $g : \{0,1\}^{\theta^{-1}\log p} \to \{0,1\}^{\log p}$ such that for any restriction σ with $|\sigma| \leqslant \log p$ we have $\operatorname{Img}(g|_\sigma) = \{0,1\}^{\log p}$.*

Proof. Let $u := \theta^{-1}\log p$ and g be random function that assigns to every $x \in \{0,1\}^u$ a value in $\{0,1\}^{\log p}$ independently and uniformly at random. We bound the probability that there exist a $y \in \{0,1\}^{\log p}$ and a restriction σ with $|\sigma| = \log p$ such that $y \notin \operatorname{Img}(g|_\sigma)$. This is easily given as follows

$$\begin{aligned}\Pr[\exists y, \sigma \ : \ y \in \{0,1\}^{\log p}, |\sigma| = \log p, y \notin \operatorname{Img}(g|_\sigma)] \leqslant \\ \leqslant p^2 \binom{u}{\log p} (1 - 1/p)^{2^{u - \log p}} \\ \leqslant p^2 (\log p)^{2\log p} e^{\log p - \frac{1}{p^2}2^u} = o(1).\end{aligned}$$

Then clearly we have that there must exist at least one function g realizing the complementary event that we bounded. Such function works also for each σ such that $|\sigma| \leqslant \log p$. □

The function $g : \{0,1\}^{\theta^{-1}\log p} \to \{0,1\}^{\log p}$ obtained from Lemma 2.1.7 can be used to define each variable z_i over $\mathbb{F}_p$ using $u = \theta^{-1}\log p$ new boolean variables $x_{i1}, \ldots, x_{iu}$:

$$z_i = \sum_{j=1}^{\log p} 2^{j-1} g_j(x_{i1}, \ldots, x_{iu}), \tag{2.1}$$

where g_j represents the j-th coordinate of g. Hence a linear equation mod p in n variables, say $\sum_i a_i z_i \equiv_p b$, can be transformed into a boolean function using equation (2.1) using $N := nu = n\theta^{-1}\log p$ boolean variables x_{ij}. Moreover if $|\operatorname{supp}(a_1, \ldots, a_n)| \leqslant d$ then the boolean encoding of that function as a CNF turns out to be a (du)-CNF. More precisely we have the following definition.

Definition 2.1.8. *Take the set $\mathscr{E} := \{E_1, \ldots, E_m\}$ of linear equations in n variables over $\mathbb{F}_p$ from Proposition 2.1.6. This can be expanded as a* CNF *over the z_i variables:*

$$\bigwedge_{i=1}^{m} E_i = \bigwedge_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} z_j \equiv_p b_i \right),$$

where E_i is the linear equation $\sum_{j=1}^{n} a_{ij} z_j \equiv_p b_i$ and $a_{ij}, b_i \in \mathbb{F}_p$. Replacing each z_j with expression given in (2.1) we obtain a boolean function

$$E_i^b := \sum_{j=1}^{n} a_{ij} \sum_{k=1}^{\log p} 2^{k-1} g_k(x_{i1}, \ldots, x_{iu}) \equiv_p b_i$$

and then the boolean function we consider is just the CNF *encoding of the following:*

$$\varphi := \bigwedge_{i=1}^{m} E_i^b.$$

*Note that since for each i we have $|\operatorname{supp}(E_i)| \leqslant p^2$, φ is a $(p^2\theta^{-1}\log p)$-*CNF *in $N := n\theta^{-1}\log p$ variables.*

Let $\mathscr{E}^b := \{E^b \ : \ E \in \mathscr{E}\}$ and $\mu(C) := \min\{|S| \ : \ S \subseteq \mathscr{E}^b \wedge S \models C\}$. We say that a clause C has *medium complexity w.r.t. μ* iff $\mu(C) \in (\frac{3}{2}\delta n, 3\delta n]$, with δ the parameter coming from Proposition 2.1.6.

The proof of the following Theorem is similar to the analogous result in [10].

Theorem 2.1.9. *Let φ and μ as above and let C be a clause over the x_{ij} variables of medium complexity w.r.t. μ then*

$$\operatorname{width}(C) \geqslant (1 - (c+1)\theta)N,$$

where c and θ are as in Proposition 2.1.6.

Proof. Let C be a clause of medium complexity, that is $\mu(C) \in (\frac{3}{2}\delta n, 3\delta n]$ and by contradiction $\operatorname{width}(C) \leqslant (1 - (c+1)\theta)N$. Take the minimal restriction ρ setting C to $\bot$, then $|\rho| = \operatorname{width}(C)$.

We say that a variable z_i is *free* if and only if $|\operatorname{dom}\rho \cap \{x_{i1}, \ldots, x_{iu}\}| \leqslant \log p$. First we prove that there are at least $c\theta n$ free variables.

Let Z be the number of z_i variables that are free. We have an upper bound for the number of x_{ij} variables non-assigned by ρ:

$$(c+1)\theta N \leqslant N - \operatorname{width}(C) \leqslant (n - Z)\log p + uZ.$$

Hence

$$c\theta N + \theta N \leqslant n\log p - Z\log p + uZ.$$

Now if $Z \leqslant c\theta n$ a contradiction follows immediately recalling that $N = un$ and $\theta N = n \log p$.

An extension of ρ to all the x_{ij} variables for i such that z_i is not free induces a restriction over the z_i variables mapping them in $\mathbb{F}_p$: let ρ^* denote such an extension. We look at it both as a restriction over the x_{ij} variables or a restriction (taking values in $\mathbb{F}_p$) over the z_i variables.

So the z_i variables that are free are exactly, by construction, the ones unfixed by ρ^*. As observed we have that the number of free variables is at least $c\theta n$ and hence $|\rho^*| \leqslant n - c\theta n = (1 - c\theta)n$.

As C is of medium complexity, there exists some set of equations $S \subseteq \mathscr{E}^b$ such that $S \models C$, $|S| \in (\frac{3}{2}\delta n, 3\delta n]$ and S is minimal w.r.t. inclusion.

This implies that for each possible ρ^* of the form described above, $\{S|_{\rho^*}\}$ is unsatisfiable. Moreover, by minimality of S, for each equation $E \in S$ there exists some ρ^* such that $E|_{\rho^*}$ is not an empty constraint.

The fact that, for each ρ^* we have that $S|_{\rho^*}$ is unsatisfiable means exactly that for all ρ^* there exists some $v \in \mathbb{F}_p^m$ (dependent on ρ^*) with $|\operatorname{supp}(v)| \leqslant |S|$ and $\sum_i v_i E_i|_{\rho^*}$ is unsatisfiable. Hence for each such ρ^* $\operatorname{supp}(\sum_i v_i E_i|_{\rho^*}) \subseteq \operatorname{dom}(\rho^*)$, otherwise we could use the variables unfixed by ρ^* to satisfy $\sum_i v_i E_i|_{\rho^*}$. Let $E^{\rho^*} := \sum_i v_i E_i|_{\rho^*}$ (where v depends on ρ^*).

Take a random linear combination of all the E^{ρ^*} for all the possible ρ^*: $\sum_{\rho^*} \alpha_{\rho^*} E^{\rho^*}$. Again we have that $\operatorname{supp}(\sum \alpha_{\rho^*} E^{\rho^*}) \subseteq \bigcup_{\rho^*} \operatorname{dom}(\rho^*)$.

Each E_i from S appears in this sum and its coefficient is uniformly random, and hence by averaging, there exists a linear combination such that at least $(1-1/p)\frac{3}{2}\delta n \geqslant \delta n$ of the E_i have non-zero coefficient. But this contradicts the expansion property as we have that $|\operatorname{supp}(\sum_{\rho^*} \alpha_{\rho^*} E^{\rho^*})| \leqslant |\bigcup_{\rho^*} \operatorname{dom}(\rho^*)| = (1 - c\theta)n$. □

Theorem 2.1.10. *For any large enough $k \in \mathbb{N}$ there exists an unsatisfiable* CNF *φ in N variables such that*

- *φ is a k-*CNF*;*
- $\operatorname{size}_{\text{treeRES}}(\varphi \vdash \bot) \geqslant 2^{(1-\widetilde{O}(k^{-1/3}))N}$.

Proof. We have that if $C, D \models E$ then $\mu(E) \leqslant \mu(C) + \mu(D)$ and hence in each possible refutation of φ there will be a clause of medium complexity. Hence from the previous Theorem we have that $\operatorname{width}(\varphi \vdash \bot) \geqslant (1 - \widetilde{O}(k^{-1/3}))N$. Then we just apply the width-size relationship from [12]. □

Corollary 2.1.11. *For every k there exists $\epsilon_k = O(k^{-1/3})$ and a sequence of unsatisfiable k-*CNF *formulas on which any DPLL algorithm runs for at least $2^{(1-\epsilon_k)n}$ steps where n is the number of variables.*

2.2 Lower Bounds for PPSZ

We first need some preliminaries as follow.

The Satisfiability Coding Lemma

Let F be a formula in conjunctive normal formal. We denote the set of the variables by $\mathrm{vbl}(F)$ and the set of satisfying assignments by $\mathrm{sat}(F)$. Given a satisfiable formula F we would like to encode an assignment $\alpha \in \mathrm{sat}(F)$ using as few bits as possible. Paturi, Pudlák and Zane [41] considered a very simple procedure. Given some ordering of the variables they assign values according to α one after another. If at some stage we can infer the value of the current variable we simply skip it go to the next variable. They considered probably the simplest inference rule one could imagine: if F contains a clause $\{x_i\}$ or $\{\overline{x}_i\}$ then the value of x_i could only be 1 or 0, respectively, given that F is satisfiable. For a formula F and a varible x and some $\alpha \in \{0,1\}$, the formula obtained from F by setting x to α is denoted by $F^{[x \mapsto \alpha]}$.

```
encode(α, π, F)
 1: β := the empty string
 2: for x_i ∈ vbl(F) in the order according to π do
 3:    if there is no clause in F consisting only of x_i or x̄_i then
 4:       append α_i to β
 5:    end if
 6:    F := F^[x_i ↦ α_i]
 7: end for
 8: return β
```

It is easy to see that one can reverse this procedure and decode a given string according to a certain permutation.

```
decode(β, π, F)
 1: α := the empty string
 2: for x_i ∈ vbl(F) in the order according to π do
 3:    if there is no clause in F consisting only of x_i or x̄_i then
 4:       append the current bit of β to α and go one bit on β ahead
 5:    else
 6:       append the inferred value of x_i to α
 7:    end if
 8:    apply the value of x_i on F
 9: end for
10: return α
```

The power of the satisfiability coding lemma is that even such a simple inference rule gives significant compression of satisfying assignments on average. To see this we need some definitions. We say that $\alpha \in \mathrm{sat}(F)$ is *sensitive on the j-th coordinate* if $F(\alpha^j) = 0$, where α^j is the same as α except that

the j-th bit is flipped. The *isolation* of α denoted by $i(\alpha)$ is the number of coordinates on which α is sensitive. We say that α is *isolated* if $i(\alpha) = n$. Assume that α is sensitive on j. This could only happen if there exists a clause in which x_j is the only true literal. We call such a clause *critical*. We are now ready to state the main result of this section.

Lemma 2.2.1 (**Satisfiability Coding Lemma** [41])**.** *Let F be a k-CNF on n variables and $\alpha \in \mathrm{sat}(F)$. We have $\mathbb{E}_\pi[|\mathsf{encode}(\alpha, \pi, F)|] \leq n - \frac{i(\alpha)}{k}$.*

Proof. We know that there are $i(\alpha)$ critical clauses for each of the sensitive variables. We pick a random permutation π. For each sensitive variable the probability that it occurs after all other variables in its corresponding critcial clause is at least $\frac{1}{k}$. Thus we have at least $\frac{i(\alpha)}{k}$ such variables on average. But notice that these are exactly those variables that are not used in the encoding. Therefore the expected length of the encoding is at most $n - \frac{i(\alpha)}{k}$. □

Despite its simplicity, the satisfiability coding lemma yields essenially the best known k-SAT algorithm and sharp depth-3 circuit lower bounds for the parity function.

PPZ and PPSZ k-SAT Algorithms

Applying the satisfiability coding lemma we can get a randomized k-SAT algorithm with expected running time significantly better than exhaustive search. The algorithm generates a random string and then it tries to decode it as if it was an encoding of a satisfying assignment.

```
ppz(F)
1: π ← a random permutation
2: β ← {0,1}^n, uniformly at random
3: α := decode(β, π, F)
4: return α
```

On a satisfiable formula F, the algorithm could produce junk. But we show that with good probability the output is a satisfying assignment.

Theorem 2.2.2 ([41])**.** *Let F be a satisfiable k-CNF on n variables. We have $\Pr[\mathsf{ppz}(F) \in \mathrm{sat}(F)] \geq 2^{-(1-\frac{1}{k})n}$.*

Proof. Let $\alpha \in \mathrm{sat}(F)$. For a permutation π, we have $\mathsf{decode}(\beta, \pi, F) = \alpha$

only if β matches on those bits that are encoded for α. Thus

$$\begin{aligned}
\Pr[\mathsf{ppz}(F)=\alpha] &= \sum_{\sigma}\Pr[\mathsf{ppz}(F)=\alpha|\pi=\sigma]\cdot\Pr[\pi=\sigma] \\
&= \mathop{\mathbb{E}}_{\sigma}[2^{-|\mathsf{encode}(\alpha,\pi,F)|}] \\
&\geq 2^{-\mathbb{E}_{\sigma}[|\mathsf{encode}(\alpha,\sigma,F)|]} \\
&\geq 2^{-n+\frac{i(\alpha)}{k}}
\end{aligned}$$

To conclude the result we need the following lemma.

Lemma 2.2.3 ([41])**.** *Let $S\subseteq\{0,1\}^n$ be a subset of the cube. For each $x\in S$ let $i(x)$ be the the number of neighbors of x in the cube that are not in S. It holds that $\sum_{x\in S}2^{-n+i(x)}\geq 1$.*

We can now write

$$\begin{aligned}
\Pr[\mathsf{ppz}(F)\in\mathrm{sat}(F)] &= \sum_{\alpha\in\mathrm{sat}(F)}\Pr[\mathsf{ppz}(F)=\alpha] \\
&\geq \sum_{\alpha\in\mathrm{sat}(F)}2^{-n+\frac{i(\alpha)}{k}} \\
&= 2^{-(1-\frac{1}{k})n}\sum_{\alpha\in\mathrm{sat}(F)}2^{\frac{-n+i(\alpha)}{k}} \\
&\geq 2^{-(1-\frac{1}{k})n}\sum_{\alpha\in\mathrm{sat}(F)}2^{-n+i(\alpha)} \\
&\geq 2^{-(1-\frac{1}{k})n}
\end{aligned}$$

and the result follows. □

A simple modification of this algorithm yields even further savings. Recall that in PPZ if at some point there is single clause we can safely infer the value of its constituent literal. For a formula F and a literal u and some parameter D we say F D-implies u if there exists a set of at most D clauses in F that imply u. We denote this by $F\models_D u$. Note that PPZ checks 1-implication. The PPSZ algorithm [40] simply considers implications with larger D and this provably turns out to be benficial. We can extend the encoding and decoding procedure to receive an extra input D. We can now state PPSZ algorithm.

Theorem 2.2.4 ([40])**.** *Let F be a k-CNF formula with a unique satisfying assignment on n variables. Then the PPSZ algorithm outputs this assignment with probability at least $2^{-(1-\epsilon_k)n}$, where $\epsilon_k\approx\pi^2/6k$.*

The same bound was obtained later with the assumption of the unique satisfying assignment removed.

```
ppsz(F, D)
1: π ← a random permutation
2: β ← {0,1}^n, uniformly at random
3: α := decode(β, π, F, D)
4: return α
```

Theorem 2.2.5 ([31])**.** *Let F be a satisfiable k-CNF formula on n variables. Then the PPSZ algorithm outputs a satisfying assignment with probability at least $2^{-(1-\epsilon_k)n}$, where $\epsilon_k \approx \pi^2/6k$.*

Recall that the success probability of the PPSZ algorithm on an input F could be written as

$$\Pr[\mathsf{ppsz}(F) \text{ succeeds}] = \sum_{\alpha \in \mathrm{sat}(F)} \mathop{\mathbb{E}}_{\pi}[2^{-|\mathsf{encode}(\alpha,\pi,F,D)|}]. \tag{2.2}$$

Therefore to lower bound this one could use Jensen's inequality. But to prove an upper bound on the success probability (and hence lower bound the expected running time) this inequality goes in the wrong direction. However if we guarantee that the encodings are large for all satisfying assignments and all permutations then we can get something. We define

$$\mathrm{codelength}(\alpha, F, D) = \min_{\pi} |\mathsf{encode}(\alpha, \pi, F, D)|$$

and

$$\mathrm{codelength}(F, D) = \min_{\alpha \in \mathrm{sat}(F)} \mathrm{codelength}(\alpha, F, D).$$

We can now upper bound the success probability as follows

$$(2.2) \leq |\mathrm{sat}(F)| \cdot 2^{-\mathrm{codelength}(F,D)}.$$

Thus to give an small upper bound of the success probability one needs to come up with formulas with few satisfying assignments and high code length. In fact we construct two different families of formulas satisfying this condition.

Hard Instances Based on Linear Systems[2]

In this section we give a probabilistic construction of a family of hard instances for PPSZ with parameter $D = n/k$ based on a linear system of equations on $\mathbb{F}_2$.

Theorem 2.2.6. *For large enough k, there exists a sequence of k-CNF formulas $\{F_n\}$ and some $\epsilon_k \in O((\ln k)^2/k)$ such that $|\mathrm{sat}(F)| \leq 2^{\epsilon_k n}$ and $\mathrm{codelength}(F, D) \geq (1-\epsilon_k)n$.*

[2]This section is based on [17]

This immediately implies a lower bound for PPSZ.

Corollary 2.2.7. *For large enough k, there exists a sequence of k-CNF formulas $\{F_n\}$ and some $\epsilon_k \in O((\log k)^2/k)$ such that* $\Pr[\texttt{ppsz}(F) \text{ succeeds}] \leq 2^{-(1-\epsilon_k)n}$.

We base our construction of hard instances on linear equations of the form $M \cdot \mathbf{x} = 0$. Construct $M \in \mathbb{F}_2^{n \times n}$ by setting each entry to 1 with probability k/n. One can write the system $M \cdot x = 0$ as a CNF formula by writing each constraint $\mathbf{m} \cdot \mathbf{x} = 0$ as an equivalent CNF formula. The following theorem holds.

Theorem 2.2.8. *Let F denote the* CNF *formula encoding the system $M \cdot \mathbf{x} = 0$. Then with high probability over the choice of M,*

$$\text{codelength}(F, D) \geqslant (1 - \epsilon_k)n$$

for some $\epsilon_k \in O((\ln k)^2/k)$.

We show how Theorem 2.2.8 implies Theorem 2.2.6 in a sequence of steps by (i) showing $M \cdot \mathbf{x} = 0$ does not have too many solutions and (ii) transforming $M \cdot \mathbf{x} = 0$ into a k-CNF formula and showing that the result is presereved.

Lemma 2.2.9. *With probability at least 0.9 over the choice of M, the system $M \cdot \mathbf{x} = 0$ has at most $2^{\epsilon_k n}$ solutions, where $\epsilon_k \in O(\ln^2(k)/k)$.*

Proof. We compute the expected number of solutions to $M \cdot \mathbf{x} = 0$. This is

$$\sum_{\mathbf{x} \in \mathbb{F}_2^n} \Pr[M \cdot \mathbf{x} = 0] \ . \tag{2.3}$$

Each summand depends only on the number of 1's in the vector $\mathbf{x}$. If $\mathbf{x}$ has exactly w 1's, then

$$\Pr[M \cdot \mathbf{x} = 0] = (\Pr[\mathcal{B}in(w, k/n) \equiv 0 \mod 2])^n \ .$$

Let p_w^n denote this probability. Thus, (2.3) is

$$\sum_{w=0}^{n} \binom{n}{w} p_w^n \ .$$

We divide this sum into two parts: for w above a certain threshold, $\mathcal{B}in(w, k/n)$ is even with probability close to $1/2$. For w below this threshold, that p_w^n might be close to 1. But there are few such $\mathbf{x}$ in the first place. We can compute p_w^n as follows.

Fact 2.2.10. $p_w^n = \frac{1+(1-2k/n)^w}{2}$.

Proof. Set $a = k/n$. We write p_w explicitely:

$$p_w^n = \sum_{0 \leqslant i \leqslant w,\ \text{even}} \binom{w}{i} a^i (1-a)^{w-i} \ .$$

Define and observe:

$$\begin{aligned} A &:= \sum_{i=0}^{w} \binom{w}{i} a^i (1-a)^{w-i} = 1 \ , \\ B &:= \sum_{i=0}^{w} \binom{w}{i} (-a)^i (1-a)^{w-i} = (1-2a)^w \ . \end{aligned}$$

In $A + B$, all terms for odd i cancel out:

$$A + B = \sum_{i\ \text{even}} \binom{w}{i} 2a^i (1-a)^{w-i} = 2p_w \ .$$

Therefore, $p_w^n = \frac{A+B}{2} = \frac{1+(1-2a)^w}{2}$. □

Fixing $w_0 := \ln(k)n/k$, above the threshold we now have

$$\begin{aligned} \sum_{w=w_0}^{n} \binom{n}{w} p_w^n &= \sum_{w=w_0}^{n} \binom{n}{w} \left(\frac{1 + (1-2k/n)^w}{2} \right)^n \\ &\leqslant 2^{-n} \sum_{w=w_0}^{n} (1 + (1-2k/n)^{w_0})^n \\ &\leqslant (1 + (1-2k/n)^{w_0})^n \\ &< (1 + e^{-2kw_0/n})^n \\ &= (1 + e^{-2\ln(k)})^n \\ &= (1 + 1/k^2)^n \leqslant e^{n/k^2} \ . \end{aligned}$$

Below the threshold, there are at most

$$\sum_{w=0}^{w_0} \binom{n}{w} \leqslant \left(\frac{en}{w_0} \right)^{w_0} \leqslant \left(\frac{ek}{\ln(k)} \right)^{\ln(k)n/k} \leqslant e^{2\ln^2(k)n/k} \ .$$

Thus both above and below the threshold the expected number of $\mathbf{x}$ such that $M \cdot x = 0$ is at most $2^{O(\ln^2(k)n/k)}$. Applying Markov's inequality the Lemma follows. □

Lemma 2.2.11. *Let F be a* CNF *formula over n variables, $T \subseteq [n]$ a set of variables, and α^* be a satisfying assignment. Set $F' := F^{[x_T \mapsto \alpha^*_T]}$. Then*

$$\text{codelength}(F', D) \geqslant \text{codelength}(F, D) - |T| \ ,$$

Proof. Let α' be a satisfying assignment of F' and π' be a permutation of $\mathrm{vbl}(F')$ achieving $|S| = \mathrm{codelength}(F', D)$, where $S = \mathsf{encode}(\alpha', \pi', F', D)$. Let α be the assignment to the variables $\mathrm{vbl}(F)$ that agrees with α' on $\mathrm{vbl}(F')$ and with α^* on the rest. Note that α satisfies F. Define a permutation π of $\mathrm{vbl}(F)$ as follows: first take the variables of S, in an arbitrary order, then the variables of $\mathrm{vbl}(F')$, according to their order in π'. Should there be leftover variables, insert them at any point. Note that encode first processes the variables in S. Thus, it will arrive at the formula F' after that. From then on, it will encode exactly $|\mathsf{encode}(\alpha', \pi', F', D)|$ variables. Therefore,

$$\begin{aligned}
\mathrm{codelength}(F, D) &\leq |\mathsf{encode}(\alpha, \pi, F, D)| \\
&\leq |S| + |\mathsf{encode}(\alpha', \pi', F', D)| \\
&\leq |T| + |\mathsf{encode}(\alpha', \pi', F', D)| \\
&= |T| + \mathrm{codelength}(F', D)
\end{aligned}$$

□

Lemma 2.2.12. *Choose $M \in \mathbb{F}_2^{n \times n}$ by setting every entry to 1 with probability k/n. Call a row* excessive *if it has more than $2ek$ 1's. With probability at least 0.9, the total number of 1's in the excessive rows is $O(nk/2^k)$.*

Proof. Fix $w \geqslant 2ek$. The probability that a row has at least w 1's is at most

$$\binom{n}{w}\left(\frac{k}{n}\right)^w \leqslant \left(\frac{en}{w}\right)^w \left(\frac{k}{n}\right)^w = \left(\frac{ek}{w}\right)^w \leqslant 2^{-w} .$$

Note that M has n rows. The expected total number of 1's in excessive rows is at most

$$\begin{aligned}
n \sum_{w=2ek}^{\infty} w \cdot 2^{-w} &= n \sum_{i=0}^{\infty} (i + 2ek) 2^{-i-2ek} \\
&= n 2^{-2ek} \left(\sum_{i=0}^{\infty} i 2^{-i} + \sum_{i=0}^{\infty} 2ek 2^{-i} \right) \\
&= n 2^{-2ek} (2 + 4ek) \in O(nk/2^k).
\end{aligned}$$

We can now apply Markov's inequality and finish the proof. □

Combining these two lemmas, we take our formula F, which is not necessarily a k-CNF formula, and fix all the variables that occur in clauses of size bigger than $2ek$ according to a satisfying assignment of F. The resulting formula F' is a $2ek$-CNF formula, and by Lemma 2.2.11,

$$\mathrm{codelength}(\alpha', F') \geqslant (1 - O(\ln^2 k/k) - O(k/2^k))n \geqslant (1 - O(\ln^2 k/k))n$$

for all satisfying assignments α' of F'. Thus, Theorem 2.2.6 follows from Theorem 2.2.8.

Proof of Theorem 2.2.8

Choose $M \in \mathbb{F}_2^{n \times n}$ by setting every entry to 1 with probability k/n. Let F denote the CNF formula encoding the system $M \cdot x = 0$. Suppose the conclusion of Theorem 2.2.8 does not hold. That is, there is a satisfying assignment β and a permutation π such that the set $S = \mathrm{EncVbl}(\beta, \pi, F, D)$, i.e., the set of encoded variables, is too small: $|S| \leqslant (1 - c \ln^2 k/k)n$ for some c to be determined later. We show that with high probability, this does not happen.

Definition 2.2.13 (Step Matrix)**.** *Let $s \in \mathbb{N}$. A matrix L is an s-step matrix if the rows $\mathbf{a}_1, \mathbf{a}_2, \ldots$ of A satisfy*

$$|\operatorname{supp}(\mathbf{a}_i) \setminus (\operatorname{supp}(\mathbf{a}_1) \cup \cdots \cup \operatorname{supp}(\mathbf{a}_{i-1}))| \in [s, 3s] \ . \tag{2.4}$$

In words, each new row introduces at least s and at most $3s$ new 1's.

We need some parameters. Define $\tau := 129 \ln^2 k/k$, $\ell := 4 \ln k$ and $w := \ln(k)n/k$.

Lemma 2.2.14 (Existence of a step matrix)**.** *Suppose $|\mathrm{EncVbl}(\beta, \pi, F, D)| \leqslant (1-2\tau)n$. Then there exists (1) a set $U \subseteq [n]$ of τn variables, (2) an n/k-step matrix L of dimension $\ell \times n$ and such that*

$$L \cdot M_U = 0 \tag{2.5}$$

where M_U is the $n \times |U|$-matrix M restricted to the variables in U.

Proof. Let β be a satisfying assignment of F and π a permutation such that $S := \mathrm{EncVbl}(\beta, \pi, F, D)$ has size $s := |S| \leqslant (1 - 2\tau)n$. By assumption, such β, π and S exist. Without loss of generality suppose $S = \{1, \ldots, s\}$ and $\pi = (1, 2, \ldots, n)$. From $F^{[\mathbf{x}_S \mapsto \beta_S]}$, ppsz can successively infer $x_{s+1}, x_{s+2}, \ldots, x_n$. This means that there are subformulas $G_{s+1}, \ldots, G_n$ of F from which ppsz infers the variables $x_{s+1}, \ldots, x_n$ such that $|G_j| \leqslant D$ for $s+1 \leq j \leq n$, and every satisfying assignment $\mathbf{x}$ of G_j with $\mathbf{x}_{<j} = \beta_{<j}$ also satisfies $x_j = b_j$. Every clause of F comes from a row of M. Thus, the subformulas $G_{s+1}, \ldots, G_n$ correspond to sets $R_{s+1}, \ldots, R_n$ of rows of M where $|R_j| \leqslant D = n/k$, and every $\mathbf{x}$ with $R_j \cdot \mathbf{x} = 0$ and $\mathbf{x}_{<j} = \beta_{<j}$ satisfies $x_j = \beta_j$.

Lemma 2.2.15. *Let M be a matrix. The unit vector $\mathbf{e}_j$ is in the row span of M if and only all solutions $\mathbf{x}$ of $M \cdot \mathbf{x} = 0$ satisfy $x_j = 0$. Furthermore, Let M be a matrix with n columns, $S \subset [n]$, $j \in [n] \setminus S$, and $\mathbf{b}$ be a solution to $M \cdot \mathbf{x} = 0$. If all $\mathbf{x}$ with $M \cdot \mathbf{x} = 0$ and $\mathbf{x}_S = \mathbf{b}_S$ satisfy $x_j = b_j$, then there is a row vector $\mathbf{m} \in \mathrm{rowspan}(M)$ such that $\operatorname{supp}(\mathbf{m}) \setminus S = \{j\}$.*

Proof. For the first part, the "only if" direction is easy: If $\mathbf{e}_j \in \mathrm{rowspan}(M)$, then there exists a row vector $\mathbf{r}$ such that $\mathbf{r} \cdot M = \mathbf{e}_j$. Now $x_j = \mathbf{e}_j \cdot x = \mathbf{r} \cdot M \cdot x = 0$. For the other direction, note that the set of solutions x is

the kernel of M, which in turn is $\text{rowspan}(M)^\perp$. Every $x \in \text{rowspan}(M)^\perp$ satisfying $x_j = 0$ means that $\mathbf{e}_j \perp \text{rowspan}(M)^\perp$, which in turn means $\mathbf{e}_j \in \left(\text{rowspan}(M)^\perp\right)^\perp = \text{rowspan}(M)$.

For the second part, take the system $M \cdot \mathbf{x} = 0$ and replace x_i by the constant b_i for each $i \in S$. This gives a new system

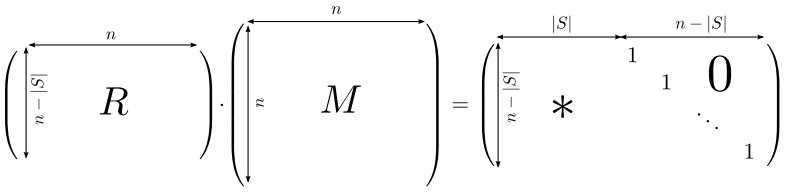
$$M' \cdot \mathbf{x}' = \mathbf{c} \tag{2.6}$$

Every solution to (2.6) satisfies $x'_j = b_j$. The restriction $\mathbf{b}' := \mathbf{b}_{[n]\setminus S}$ is a solution to (2.6). If $\mathbf{y}$ is a solution to $M' \cdot \mathbf{x}' = 0$, then $\mathbf{y} + \mathbf{b}'$ is a solution to (2.6), and therefore $y_j + b_j = b_j$. So every solution $\mathbf{y}$ of $M' \cdot \mathbf{x}' = \mathbf{c}$ satisfies $y_j = 0$. By the first part, $\mathbf{e}_j \in \text{rowspan}(M')$. So there exists a row vector $\mathbf{r}$ such that $\mathbf{r} \cdot M' = \mathbf{e}_j$. Comparing $\mathbf{m} := \mathbf{r} \cdot M$ with $\mathbf{r} \cdot M' = \mathbf{e}_j$? we not that it has additional coordinates: $i \in S$. However, the coordinates $i \in [n] \setminus S \setminus \{j\}$ are still 0. Thus, $\text{supp}(m) \setminus S = \{j\}$. □ □

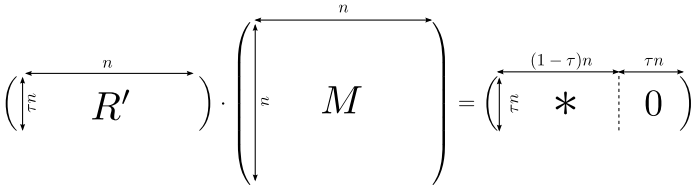
The lemma implies that there are sets $R_{s+1}, \ldots, R_n$ of rows of M such that $|R_j| \leqslant D$ and

$$\sum_{i \in R_j} \mathbf{m}_i = (*, \ldots, *, 1, 0, \ldots, 0) \ , \forall j : s+1 \leq j \leq n$$

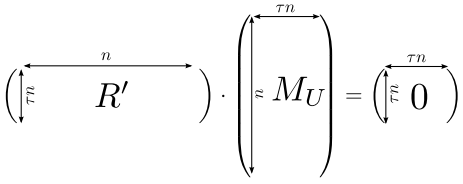
where the 1 is at the j^{th} position and $*$ mean either of 0 or 1. Here, $\mathbf{m}_i$ is the i^{th} row of M. If we let $\mathbf{r}_i$ denote the characteristic vector of R_j, then $\sum_{i \in R_j} \mathbf{m}_i = \mathbf{r}_j \cdot M$. Let R denote the $(n-s) \times n$-matrix with rows $\mathbf{r}_j$, $s+1 \leqslant j \leqslant n$. We see that the matrix M satisfies the following equation:



Every row of R contains at most D 1's. Note that the upper triangle at the end of $R \cdot M$ implies that $R \cdot M$ is full rank, and R has full rank $n - s \geqslant 2\tau n$, too. Let us look at the "upper half" of that equation, that is, let R' consist of the τn first rows of R. R' has rank τn and satisfies the following equation:

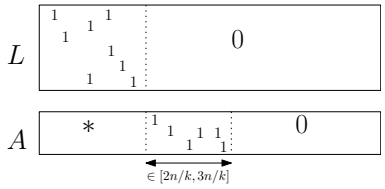


Finally, let $U \subseteq [n]$ be the last τn variables, and M_U the matrix M restricted to the variables in U. We get:



We can now construct the step matrix L. We successively build L by adding rows to it. Each such row will be a linear combination of rows of R'. This guarantees that $L \cdot M_U = 0$. Suppose we have built up parts of L and want to find a new row we can add to it. For this we build a temporary matrix A.

Grow a matrix A with n columns by adding rows of R' to it. Note that R' has full rank, therefore $|\operatorname{supp}(R')| \geqslant \tau n$. As long as $|\operatorname{supp}(L)| \leqslant \tau n - 2n/k$, since $|\operatorname{supp}(R')| \geqslant \tau n$ there is a point in time where $|\operatorname{supp}(A) \setminus \operatorname{supp}(L)| \geqslant 2n/k$ for the first time. At this point, $|\operatorname{supp}(A) \setminus \operatorname{supp}(L)| \leqslant 3n/k$, since the last row adds at most $D = n/k$ 1's.



Take a random linear combination of the rows in A. This has at most $3n/k$ 1's outside $\operatorname{supp}(L)$ and, on expectation, at least n/k 1s outside $\operatorname{supp}(L)$. Thus, there is some linear combination $\mathbf{r}$ with $n/k \leqslant |\operatorname{supp}(\mathbf{r}) \setminus \operatorname{supp}(L)| \leqslant 3n/k$. Add this row to L.

Since $|\operatorname{supp}(L)|$ is 0 at the beginning and grows by at most $3n/k$ in each step, we can grow L for at least

$$\frac{|\operatorname{supp}(R')|}{3n/k} \geqslant \frac{\tau k}{3} \geqslant \frac{129 \ln(k)}{3} \geqslant 4 \ln(k) \ .$$

steps. This final matrix L satisfies $L \cdot M_U = 0$, since every row of L is a linear combination of rows of R'. The dimensions of L are $4 \ln(k) \times n$, and it is an n/k-step matrix. This finishes the proof.

□

The remaining rather straightforward propositions show that it is very unlikely to have set U and matrices L as in Lemma 2.2.14 and thus proving the result.

Proposition 2.2.16. *There are at most* $\binom{n}{\tau n}$ *ways to choose* U*, at most* $\binom{n}{3\ell n/k}^{\ell}$ *ways to choose* L*.*

Proof. U is a set of τn variables, so there are $\binom{n}{\tau n}$ ways to choose U. As for L, it is an n/k-step matrix of dimensions $\ell \times n$. This means that every row introduces at most $3n/k$ new 1's. Thus, every row has at most $3\ell n/k$ 1's. There are $\binom{n}{3\ell n/k}$ ways to choose such a row, and ℓ such rows to choose. □ □

Proposition 2.2.17. *For each fixed U and step matrix L*

$$\Pr[L \cdot M_U = 0] \leqslant e^{-\ell\tau n/2} ,$$

where the probability is taken over the choice of M.

Proof. M_U has τn columns, each of which is chosen independently. Thus, it suffices to show that

$$\Pr[L \cdot \mathbf{m} = 0] \leqslant e^{-\ell/2} \tag{2.7}$$

for every column $\mathbf{m}$ of M. Let $\mathbf{r}_1, \ldots, \mathbf{r}_\ell$ be the rows of L. We prove that for $1 \leqslant i \leqslant \ell$ it holds that

$$\Pr[\mathbf{r}_i \cdot \mathbf{m} = 0 \mid \forall\ i' < i,\ \mathbf{r}_{i'} \cdot \mathbf{m} = 0] \leqslant e^{-1/2} . \tag{2.8}$$

From here, (2.7) follows by the chain rule of conditional probabilities.

To show (2.8), write $\mathbf{r}_i = \mathbf{r}_{\text{old}} + \mathbf{r}_{\text{new}}$, where $\mathbf{r}_{\text{new}}$ contains exactly those 1's that are not in $\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}$. Since L is an n/k-step matrix, $|\operatorname{supp}(\mathbf{r}_{\text{new}})| \geqslant n/k$. Denote by $\mathcal{E}$ the condition that $\mathbf{r}_{i'} \cdot \mathbf{m} = 0$ for all $i' < i$. Then

$$\begin{aligned} \Pr[\mathbf{r}_i \cdot \mathbf{m} = 0 \mid \mathcal{E}] &= \Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} + \mathbf{r}_{\text{old}} \cdot \mathbf{m} = 0 \mid \mathcal{E}] \\ &= \Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = \mathbf{r}_{\text{old}} \cdot \mathbf{m} \mid \mathcal{E}]. \end{aligned} \tag{2.9}$$

Since $\mathbf{r}_{\text{new}}$ only contains 1's that are not present in any previous $\mathbf{r}_{i'}$, the random variable $\mathbf{r}_{\text{new}} \cdot \mathbf{m}$ is independent of $\mathbf{r}_{\text{old}} \cdot \mathbf{m}$ and $\mathcal{E}$. Define $y := \mathbf{r}_{\text{old}} \cdot \mathbf{m}$ and estimate

$$\Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = y] .$$

We write $m = |\operatorname{supp}(\mathbf{r}_{\text{new}})|$ and $p = k/n$. Note that

$$\Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = y] = \begin{cases} \Pr[\mathcal{B}in(m,p) \text{ is even}] & \text{when } y = 0 \\ \Pr[\mathcal{B}in(m,p) \text{ is odd}] & \text{when } y = 1 \end{cases}$$

The probability that $\mathcal{B}in(m,p)$ is even is:

$$\begin{aligned} \Pr[\mathcal{B}in(n/k, k/n) \text{ is even}] &= \frac{1 + (1 - 2k/n)^{n/k}}{2} \\ &\leqslant \frac{1 + e^{-2}}{2} \leqslant e^{-1/2} . \end{aligned}$$

The probability that $\mathcal{B}in(m,p)$ is odd is at most $1/2$. Thus, we see that $\Pr[\mathbf{r}_{\text{new}} \cdot \mathbf{m} = y] \leqslant e^{-1/2}$, no matter what y is. This concludes the proof. □ □

We combine these two propositions:

Proposition 2.2.18. *The probability that there exists a set U and matrices L and Z as described in Lemma 2.2.14 is at most*

$$\binom{n}{\tau n} \cdot \binom{n}{4\ell n/k}^{\ell} \cdot e^{-\ell\tau n/2} .$$

For our choice of parameters, this is $o(1)$.

Proof. The expression follows from the two previous propositions via a union bound. Let us evaluate it for our choice of parameters. Recall that $\tau = 129\ln^2(k)/k$, and $\ell = 4\ln(k)$. Thus

$$\begin{aligned}
\binom{n}{\tau n} \cdot \binom{n}{4\ell n/k}^{\ell} \cdot e^{-\ell\tau n/2} &\leqslant \left(\frac{ek}{129\ln^2(k)}\right)^{\tau n} \cdot \left(\frac{ek}{4\ell}\right)^{\frac{4\ell^2}{k}} \cdot e^{-\ell\tau n/2} \\
&= \left(\frac{ek}{129\ln^2(k)}\right)^{129\frac{\ln^2(k)}{k}n} \cdot \left(\frac{ek}{16\ln(k)}\right)^{\frac{64\ln^2(k)}{k}} \\
&\quad \cdot e^{-358\ln^3(k)n/k} \\
&= o(1)
\end{aligned}$$

□

There is a stronger version of PPSZ which does resolution of bounded width in each round instead of taking subformulas. In the paper we actually consider such an inference heuristic and show that the same construction is hard also for those algorithms.

Hard Instances Based on Digraphs[3]

In this section we prove a strong exponential lower bound for PPSZ when $D = o(\log\log n)$. Let $G = (V, E)$ be a digraph in which every vertex has out-degree at most k. For each vertex x with out-neighbors $y_1, \ldots, y_k$ we make the clause $\{x, \bar{y}_1, \ldots, \bar{y}_k\}$ and consider the CNF F formed by taking all these clauses. It is easy to characterize the set of satisfying assignments of F

$$\mathrm{sat}(F) = \{\alpha \mid \alpha^{-1}(0) \text{ induces a subgraph with no sinks}\}$$

Let π be a permutation of the variables (vertices) and let S be the set of variables encoded by $\mathsf{encode}(1, \pi, F, D)$.

[3]This section is based on an unpublished manuscript of Dominik Scheder and Navid Talebanfard.

Lemma 2.2.19. *$F^{[S\mapsto 1]}$ has a unique solution, the all-one assignment. Furthermore, $G - S$ is acyclic.*

Proof. S is the set of variables encoded in $\mathsf{encode}(1, \pi, F, D)$. That means, from $F^{[S\mapsto 1]}$ one can infer the values of all other variables using some "obvious implication" rule. In particular, $F^{[S\mapsto 1]}$ implies the value of all other variables. Thus, there is exactly one satisfying assignment: The all-one assignment.

For the second claim, note that if $G - S$ is not acyclic, then there are at least two satisfying assignments of $F^{[S\mapsto 1]}$: The all-one assignment, and the assignment setting the vertices on the cycle to 0 and everything else to 1. □

Definition 2.2.20. *Let $G = (V, E)$ be a directed graph. A* decycling set *is a set $S \subseteq V$ such that $G - S$ is acyclic.*

Thus if we have a linear lower bound on the size of decycling sets in G we can get a lower bound on the code length of the all-one assignment. But this is not sufficient to get a lower bound on the success probability of PPSZ, we need to give lower bounds for all the satisfying assignments. Since the set of zeros in a satisfying assignment contains a cycle, if we somehow "hit" all of the cycles by adding certain clauses and forcing them to have at least one true variable, then only the all-one assignment survives. This should however be done in a careful way to keep the code length high enough.

The new CNF F' has three parts each being a set of clauses. The first part is just F as above derived from a digraph. The second part is a set P of clauses of the form $\{x, y\}$ where x and y are vertices in the digraph, and the last part Q consists of some singleton clauses $\{x\}$.

Lemma 2.2.21. *Let $F' = F \cup P \cup Q$ be a CNF as above derived from a digraph $G = (V, E)$ and let D be the ppsz parameter. If for some integers m, z and $s > t > D$*

1. *each decycling set in G has size at least s,*
2. *there are at most m cycles of length at most t,*
3. *for each cycle C of length greater than t, there exists a pair $\{u, v\} \in P \cap C$,*
4. *the number of pairs $\{u, v\} \in P$ for which $d(u, v) \leq D$ is z,*
5. *and for each cycle of length at most t, there exists some $x \in Q \cap C$,*

then the all-one assignment is the only satisfying assignment of F' and it has code length at least $s - mt - z - |Q|$.

Proof. We first notice that the all-one assignment is the only surviving assignment since in every cycle there is at least one true variable. Let X be the set of variables encoded by $\mathsf{encode}(1, \pi, F')$. We will prove that $|X| \geq s - mt - z - |Q|$. Let P' be the set of all pairs in P that are of distance at most D in the digraph. Therefore $z = |P'|$. We move all the vertices in Q and all vertices appearing in P' to X to get X'. If X' is a decycling set then by assumption we have $|X| + |Q| + z \geq s$ and thus $|X| \geq s - z - |Q|$ and we would be done. Therefore assume that there are cycles in $V \setminus X'$. We then move all cycles of length at most t to X'. If the resulting set is decycling then we would again be done, since there are m cycles of length at most t we get $|X'| + |Q| + z + mt \geq s$ and thus $|X| \geq s - mt - z - |Q|$.

Let Y be the of vertices appearing in some cycle of length at most t in $V \setminus X'$. We shall now assume that we have a cycle C in $Z = V \setminus (X' \cup Y)$ of length bigger than t, say $C = \{v_1, \ldots, v_r\}$ with $r > t$. Let C be a smallest such cycle. We then move all the vertices in $Z \setminus C$ to the encoding and consider $T = F'^{[X' \cup Y \cup (Z \setminus C) \mapsto 1]}$. Assume that under π we have $v_1 \prec \ldots \prec v_r$. Therefore T D-implies v_1. This means that there exists a set of clauses $S \subseteq T$ with $|S| \leq D$ such that $S \models v_1$. By minimality of C we know that there are no edges between v_i's other than those belonging to the cycle. The clauses in T are then $\{v_1, \bar{v}_2\}, \{v_2, \bar{v}_3\}, \ldots, \{v_{r-1}, \bar{v}_r\}, \{v_r, \bar{v}_1\}$ corresponding to the edges in C, and a set of pairs from P. Assume that S does not contain any clause from P and thus contains only some of the $\{v_i, \bar{v}_{i+1}\}$ clauses corresponding to the edges in C. At least one of these clauses should contain v_1, since S forces the value of v_1. Now take a maximal path formed by edges in S containing v_1. We set the vertices in this path to 0 and every other vertex in C to 1. This assignment clearly satisfies S which shows that $S \not\models v_1$, a contradiction. Therefore assume that S contains some pair $\{v_i, v_j\} \in P$. We again take a maximal path R containing v_1. If there is no P-pair between any of the vertices on this path, we can apply the same assignment get to a contradiction. We thus assume that there is P pair on this path. But since the P-pairs have distance bigger than D, this means $|R| \geq D$. This path together with the P-pair imply that S contains at least $D + 1$ clauses which is a contradiction and thus the result follows. □

It is now clear how to get a hard instance for PPSZ based on digraphs. We just need to prove the existence of an F' as above with parameters $s = (1 - o_k(1))n$, $mt = o(n)$, $z = o(n)$ and $|Q| = o(n)$.

Lemma 2.2.22. *Build a directed graph G by choosing k out-edges for each vertex, uniformly with replacement, and independently and form F based on G as above. Next for each pair of vertice x and y add the clause $\{x, y\}$ with probability $p = \Theta(\frac{\log^2 k}{\log n})$, and finally for each cycle of length at most t pick some arbitrary vertex x in the cycle and add the clause $\{x\}$ to get Q. Then*

with probability $1-o(1)$ *the resulting formula has parameters* $s = (1-o_k(1))n$, $mt = o(n)$, $z = o(n)$ *and* $|Q| = o(n)$.

Proof. We prove this in stages.

Claim 1. *With probability* $1-o(1)$, *every decyclinig set in* G *has size at least* $(1 - O(\log k/k))n$.

Proof. Let us bound from above the probability that the random graph G has a decycling set of size s. We give a rough estimate, not claiming to be close to the truth. If $G - S$ is acyclic, we can partition the remaining $r := n - s$ vertices into two sets T_1, T_2 such that there is no edge from T_1 into T_2. Let us fix sets S, T_1, T_2 of size s, $r/2$, and $r/2$, respectively. What is the probability that no edge goes from T_1 to T_2? For every vertex, we choose k out-edges. We see that the probability we are looking for is

$$\left(1 - \frac{r}{2n}\right)^{k\cdot r/2} .$$

We take a union over all choices of S, T_1, T_2: There are $\binom{n}{s} = \binom{n}{r}$ ways to choose S; given that, there are $\binom{r}{r/2} \leqslant 2^r$ ways to choose T_1, T_2. Thus, the probability that G has a decycling set of size s is at most

$$\binom{n}{r} 2^r \left(1 - \frac{r}{2n}\right)^{k\cdot r/2} < \left(\frac{en}{r}\right)^r 2^r e^{\frac{kr^2}{4n}} = \left(\frac{2e}{\tau} \cdot e^{-k\tau/4}\right)^r \tag{2.10}$$

for $\tau := r/n$. This is $o(1)$ if and only if $\frac{2e}{\tau} \cdot e^{-k\tau/4} < 0$, which holds if $k \geqslant c \cdot \frac{1}{\tau} \ln \frac{1}{\tau}$. So if we choose $\tau := \frac{\log k}{Ck}$ for some constant C, then with probability $o(1)$, G has a decycling set of size $(1-\tau)n$. □

Claim 2. *With probability* $1 - o(1)$ *there are at most* $\sqrt{n}$ *cycles of length* $\Theta(\log n)$.

Proof. Let X_ℓ denote the number of cycles of length ℓ. By Markov's inequality it is sufficient to bound $\mathbb{E}[X_\ell]$ for $\ell = \Theta(\log n)$.

$$\begin{aligned}
\mathbb{E}[X_\ell] &= \frac{\ell!}{\ell}\binom{n}{\ell}\left(\frac{k}{n}\right)^\ell \\
&\leq \frac{\ell!}{\ell}\frac{n^\ell}{\ell!}\left(\frac{k}{n}\right)^\ell \\
&= \frac{k^\ell}{\ell} \\
&\leq k^\ell
\end{aligned}$$

Setting $\ell = \log n/4\log k$ then gives $\mathbb{E}[X_\ell] \leq n^{1/4}$ and thus $\Pr[X_\ell \geq \sqrt{n}] \leq n^{-1/2}$ proving the claim. □

Claim 3. *With probability* $1 - o(1)$ *every cycle of length* $\Omega(\log n)$ *is hit by* P*, i.e., for every long enough cycle* C *there exist* $u, v \in C$ *such that* $\{u, v\} \in P$*.*

Proof. The probability that a cycle of length ℓ is not hit by P is easily computed by

$$(1-p)^{\binom{l}{2}} \leq e^{-p\binom{l}{2}}$$

Thus the expected number of cycles not hit by P is at most

$$k^{\ell} e^{-p\binom{l}{2}}$$

But by the choice of p this is clearly $o(1)$ and the claim follows. □

Claim 4. *With probabiliy* $1 - o(1)$ *there are at most* $o(n)$ *pairs* $\{u, v\} \in P$ *such that* $d(u, v) \leq D$*.*

Proof. For $\ell \leq D$, the expected number of pairs $\{u, v\} \in P$ such that there exists a path of length ℓ from u to v is

$$p\binom{n}{\ell+1}(\ell+1)!\left(\frac{k}{n}\right)^{\ell} \leq p \cdot k^{\ell} \cdot n$$

Therefore the expected number of pairs of distance at most D in P is at most

$$D \cdot p \cdot k^{D} \cdot n$$

For $D = o(\log \log n)$ this is $o(n)$ and we are done. □

□

Chapter 3

Variable Space versus Depth in Resolution[1]

3.1 introduction

Resolution (RES) is one of the most fundamental and extensively studied proof systems, using which one can refute unsatisfiable CNF formulas applying the following inference rule

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

where C and D are disjunctions of literals and x is any variable. Every resolution refutation induces a DAG in the following way. There is a node for each clause appearing in the proof, and every such node will be connected by an edge to the nodes corresponding to the two clauses from which this clause was derived. The most basic parameter of a resolution is its *size*, for which proving exponential lower bounds would be evidence for NP $\neq$ co-NP. There is a large body of work studying size in resolution (see e.g., [28], [52], [18], [9]). But other parameters such as width and space have also been extensively studied (see e.g. [12], [3], [39]). But only in 2011, Urquhart [53] initiated the study of resolution depth, that is the longest path from an axiom to the conclusion of the proof. He showed that depth is lower bounded by width and variable space. He also demonstrated a strong separation between depth and width by constructing formulas that have $O(1)$ width refutations, but require $\Omega(n/\log n)$ depth. Urquhart asked if it is possible to obtain such separations between depth and variable space. In this chapter we show that for formulas with refutations of small variable space, there exist refutations of small depth. More precisely we show that if a formula has a refutation of variable space $s = O(\log \log n)$, it also has a refutation of depth $O(s^2 \log n)$. This shows that at least for small ranges of s, strong separations between depth and variable space are not possible.

[1]This chapter is based on an ongoing work of Ilario Bonacina and Navid Talebanfard.

Preliminary Definitions

We define a *configuration* to be a set of clauses and hence a CNF. We can think of a resolution refutation of a formula φ as a sequence $(\mathcal{C}_1, \ldots, \mathcal{C}_t)$ of configurations such that $\mathcal{C}_1 = \emptyset$ and $\bot \in \mathcal{C}_t$ and $\mathcal{C}_i$s are obtained in one the following ways:

Axiom download $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{D\}$, where D is a clause in φ.

Resolution inference $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{C \vee D\}$, where $C \vee D$ is a obtained from $\mathcal{C}_i$ by applying resolution rule to some $C \vee x$ and $D \vee \neg x$ in $\mathcal{C}_i$.

Erasure $\mathcal{C}_{i+1} = \mathcal{C}_i \setminus \{D\}$.

We define the *variable space* of this proof be the maximum number of distinct variables appearing in $\mathcal{C}_i$s. The variable space of φ denoted by $\mathsf{VSpace}(\varphi \vdash \bot)$ is the smallest variable space over all resolution refutations of φ. The *depth* of the proof is the length of the longest path from $\bot$ to a clause in the proof. The depth of φ is the smallest depth over all resolution refutations of φ, and we denote this by $\mathsf{Depth}(\varphi \vdash \bot)$. We distinguish between two different kinds of resolution proofs, *syntact* and *semantic*. A syntactic proof is just as described above. A semantic proof however can infer any clause at each step, which is logically implied by the current configuration. We denote the semantic variable space by $\mathsf{VSpace}_{sem}(\varphi \vdash \bot)$. Note that a syntactic proof is also semantic, therefore $\mathsf{VSpace}_{sem}(\varphi \vdash \bot) \leq \mathsf{VSpace}(\varphi \vdash \bot)$.

The Interrogation game was defined by Urquhart [53] to characterize depth in resolution. Given an unsatisfiable formula φ, two players, the Interrogator and the Witness play the following game. Each round starts with the Interrogator picking a variable x not already chosen. The Witness then has to assign a value to x and restricting the formula accordingly. The game finishes when some clause of φ is falsified. We say that the Interrogator wins that $\mathsf{DepthGame}(\varphi, k)$, if he can force to the Witness to falsify a clause in at most k steps.

Theorem 3.1.1 (Urquhart [53])**.** *Let φ is an unsatisfiable* CNF*, then Interrogator wins* $\mathsf{DepthGame}(\varphi, k)$ *if and only if* $\mathsf{Depth}(\varphi) \leqslant k$*.*

3.2 Main Result

Proposition 3.2.1. *Let φ an unsatisfiable* CNF *and π a semantical Resolution proof of φ. Then* $\mathsf{Depth}(\varphi \vdash \bot) \leqslant \mathsf{VSpace}(\pi) \log(\mathsf{Length}(\pi))$*.*

Proof. Let $\pi = (\mathcal{C}_1, \ldots, \mathcal{C}_t)$ and $\kappa := \mathsf{VSpace}(\pi) \log t$. By Theorem 3.1.1 it is enough to give a strategy for the Interrogator to win $\mathsf{DepthGame}(\varphi, \kappa)$.

A winning strategy for the Interrogator is the following: maintain two integers a_j and b_j, such that the Witness is satisfying $\mathcal{C}_{a_j}$ and falsifying $\mathcal{C}_{b_j}$.

At the beginning of the game $a_0 = 1$ and $b_0 = t$. At step j the Interrogator queries all the variables in $\mathcal{C}_{\lfloor \frac{a_j+b_j}{2} \rfloor}$ and update a_j or b_j according to the answer of the Witness: if the answer (together with the previous answers) is falsifying $\mathcal{C}_{\lfloor \frac{a_j+b_j}{2} \rfloor}$ then $b_{j+1} = \lfloor \frac{a_j+b_j}{2} \rfloor$ otherwise $a_{j+1} = \lfloor \frac{a_j+b_j}{2} \rfloor$. At each step the Interrogator queries at most $\mathsf{VSpace}(\pi)$ variables and after at most $\tau = \log t$ steps a_τ and b_τ are two consecutive integers. We have to show that at step τ the Witness necessarily loses. We have that either $\mathcal{C}_{a_\tau} \models \mathcal{C}_{b_\tau}$ or $\mathcal{C}_{b_\tau} = \mathcal{C}_{a_\tau} \cup \{C\}$ where $C \in \varphi$.

In the first case (that is $\mathcal{C}_{a_\tau} \models \mathcal{C}_{b_\tau}$) $\mathcal{C}_{a_\tau}$ is satisfied and $\mathcal{C}_{b_\tau}$ is falsified: it follows that the Witness has played inconsistently.

In the second case, that is $\mathcal{C}_{b_\tau} = \mathcal{C}_{a_\tau} \cup \{C\}$ where $C \in \varphi$, since $\mathcal{C}_{a_\tau}$ is satisfied by the Witness, the only way to falsify $\mathcal{C}_{b_\tau}$ would be to falsify C and hence finishing the game. □

Proposition 3.2.2. *Let φ and unsatisfiable* CNF *in n variables, π a semantic Resolution proof of φ and $s = \mathsf{VSpace}(\pi)$. Then there exists a proof with the same variable space of length at most $n^s 2^{2^s}$.*

Proof. Let $\pi = (\mathcal{C}_1, \ldots, \mathcal{C}_t)$. If for some $i \neq j$ we have that $\mathcal{C}_i$ and $\mathcal{C}_j$ are semantically equivalent, we can simply remove $\mathcal{C}_i, \ldots, \mathcal{C}_{j-1}$ and still have a valid refutation of φ having the same variable space. We repeat this until we have no repetitions in the proof.

Then we just bound the number of possible memory configurations: Each memory configuration is a Boolean function on s variables represented by a CNF. There are $\binom{n}{s}$ ways to pick the variables and 2^{2^s} distinct boolean functions we can have on those variables. Hence we have the bound that $t \leqslant n^s 2^{2^s}$, using the trivial estimation $\binom{n}{k} \leqslant n^k$. □

Putting together Propositions 3.2.1 and 3.2.2 we have immediately the following corollary.

Corollary 3.2.3. *Let φ an unsatisfiable* CNF *in n variables and $s = \mathsf{VSpace}_{sem}(\varphi \vdash \bot)$. If $s = O(\log \log n)$ then $\mathsf{Depth}(\varphi \vdash \bot) = O(s^2 \log n)$.*

Proof. Putting together Propositions 3.2.1 and 3.2.2 we have that

$$\mathsf{Depth}(\varphi \vdash \bot) \leqslant s \log(n^s 2^{2^s}) = s(s \log n + 2^s),$$

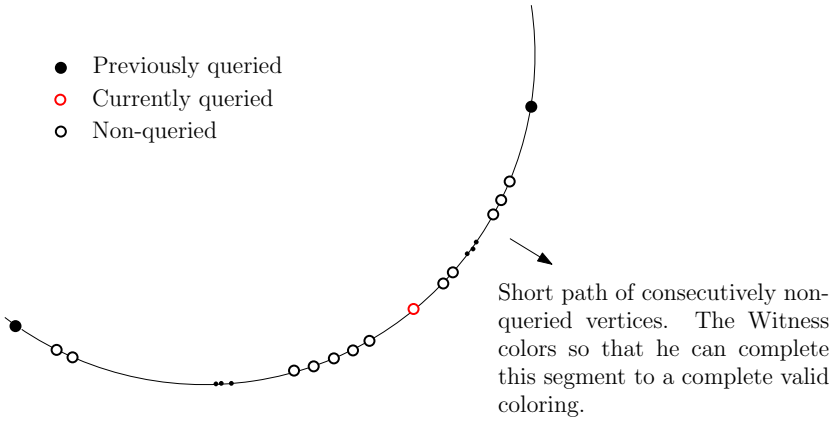
hence, if $s = O(\log \log n)$ then $\mathsf{Depth}(\varphi \vdash \bot) = O(s^2 \log n)$. □

We complement our result with a lower bound for depth showing that there are formulas refutable in constant variable space which require logarithmic depth.

Theorem 3.2.4. *For any odd n, there exists an unsatisfiable 2-*CNF *formula refutable by keeping 3 variables in each memory configuration, but any resolution refutation of it requires depth $\Omega(\log n)$.*

Proof. Consider the cycle graph on vertices $x_1, \ldots, x_n$. Formula F expresses that this graph is 2-colorable, that is for each edge $(x_i, x_{(i+1 \mod n)})$ we put two clauses $(x_i \vee x_{(i+1 \mod n)})$ and $(\overline{x}_i \vee \overline{x}_{(i+1 \mod n)})$. Since n is odd, this is obviously unsatisfiable. A resolution refutation could be obtained by starting from any edge and resolving the corresponding clauses with that of the next adjacent edge, and continuing this fashion. It is easy to see that one only needs to keep 3 variables in memory each time.

To prove the depth lower bound, we need to show that the Interrogator cannot win $\mathsf{DepthGame}(F, \Omega(\log n))$, that is we show that there exists a strategy for the Witness that enables him to continue playing for at least $\Omega(\log n)$ steps. The strategy is as follows. When the Interrogator queries the first variable, the Witness answers arbitrarily. From this step on when a variable is queried, the Interrogator finds a previously queried variable connected to the current variable by a shortest sequence of consecutively non-queried variables. Then he colors this vertex so that it is possible to extend the coloring to get a valid coloring of this path.



□

The best response to this strategy for the Interrogator is to always query a vertex which is almost equally far apart from two already queried variables. But even in this case the Witness can play for at least $\Omega(\log n)$ steps.

Chapter 4

On the Structure and the Number of Prime Implicants of k-CNF Formulas

A *prime implicant* of a Boolean function f is a maximal subcube contained in $f^{-1}(1)$. It is known that among Boolean functions on n variables the maximum number of prime implicants is between $\Omega(\frac{3^n}{n})$ and $O(\frac{3^n}{\sqrt{n}})$ (see [16]). It is interesting to give finer bounds for restricted classes of functions. This problem has indeed been studied for DNF with a bounded number of terms. It is known that a DNF with k terms has at most $2^k - 1$ prime implicants (see e.g. [50]).

In this chapter we consider this problem for the class of k-CNF functions and prove that the number of prime implicants in a k-CNF formula on n variable in which every variable appears at most r times is at most $3^{(1-\frac{1}{rk})n}$. Indeed understanding the structure of the set of satisfying assignments of k-CNF formulas has been a crucial subject in computational complexity, in particular in developing k-SAT algorithms and bounded depth circuit lower bounds. Notable examples are the characterization of satisfying assignments of a 2-CNF which yields a polynomial time algorithm for 2-SAT (see e.g. [2]), and the Satisfiability Coding Lemma of Paturi, Pudlák and Zane [41]. In fact we obtain our main result by a generaliztion of this lemma.

To state our results, we need a few definitions as follows. A *restriction* on a set of variables X is a mapping $\rho : X \to \{*, 0, 1\}$. We call a variable *free* if it is assigned $*$, and we call it *fixed* otherwise. The *size* of ρ is defined to be the number of variables fixed to 0 or 1 and we denote it by $|\rho|$. A *sub-restriction* ρ' of ρ is one such that $\rho(x) = *$ implies $\rho'(x) = *$ and we denote this by $\rho' \sqsubseteq \rho$. We also define $\rho^{[x \to *]}$ to be the restriction obtained from ρ by unspecifying x. For a function f, we define $f|_\rho$ to be the subfunction after setting values to the fixed variables according to ρ. An *implicant* of f, is any restriction ρ such that $f_\rho \equiv 1$. Furthermore, if unspecifying any fixed variable

does not yield the constant 1 function, we say that ρ is a *prime implicant* of f. If a prime implicant leaves some variable free we call it *partial*. We denote the set of prime implicants of F by $\mathsf{PI}(F)$. A read-r formula is one in which every variable appears at most r times. The set of variables of a formula F is represented by $\mathrm{vbl}(F)$.

A (d,k)-CSP is set of constraints each on at most k variables, where every variable can take a value from a set of size d. For example a $(2,k)$-CSP is just a k-CNF. We say that an assignment α is an *isolated solution* for a CSP F, if α satisfies F and changing the value of any single variable, falsifies some constraint. In what follows we give a generalization of the Satisfiability Coding Lemma to bound the number of isolated solutions in a (d,k)-CSP.

We have two main results in this section. Using an old characterization of satsifying assignments of 2-CNFs, we give an essentially tight bound for the number of prime implicants of 2-CNFs. For k-CNFs in general with $k \geq 3$, we obtain a bound assuming that every variable appears in at most a bounded number of clauses.

4.1 $k = 2$[1]

In this section we prove sharp bounds for the number of prime implicants of 2-CNF formulas.

Theorem 4.1.1. *For every n, there exists a 2-CNF formula F on n variables with $|\mathsf{PI}(F)| \geq 3^{\frac{n}{3}}$.*

Proof. Let $n = 3m$ and consider the following formula on variable set $\{x_1, \ldots, x_m, y_1, \ldots, y_m, z_1, \ldots, z_m\}$ suggested to us by Dominik Scheder:

$$T(x,y,z) = \bigwedge_{i=1}^{m} (x_i \vee y_i) \wedge (y_i \vee z_i) \wedge (x_i \vee z_i).$$

It is easy to see that every prime implicant of T and every $1 \leq i \leq m$ must set exactly two variables among x_i, y_i and z_i to 1. Therefore $T(x,y,z)$ has $3^{\frac{n}{3}}$ prime implicants.

□

Theorem 4.1.2. *For any 2-CNF F on n variables, we have $|\mathsf{PI}(F)| \leq (1 + o(1))3^{\frac{n}{3}}$.*

Let F be a 2-CNF on $\{x_1, \ldots, x_n\}$ and let ρ be a prime implicant that fixes all the variables. We claim that ρ is an *isolated* satisfying assignment for F, that is if we change the value of any single one of the variables, the formula evaluates to 0. To see this note that if changing the value of some variable x_i still satisfies F, we can simply unspecify x_i and get a smaller restriction which

[1] This section is based on [51]

yields the constant 1 function, contradicting the minimality of ρ. We can now apply the Satisfiability Coding Lemma and bound the number of such prime implicants by $2^{\frac{n}{2}}$.

Lemma 4.1.3 (The Satisfiability Coding Lemma [41])**.** *Any k-CNF on n variables has at most $2^{(1-\frac{1}{k})n}$ isolated satisfying assignments.*

It thus remains only to bound the number of partial prime implicants. We need some terminology. We define the *implication digraph* of F which we denote by $D(F)$ as follows. The vertex set consists of all literals $x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n$. For every clause $x \vee y$ in F we put two directed edge $\overline{x} \to y$ and $\overline{y} \to x$ in $D(F)$. If there is a clause consisting of only one literal x, we add the edge $\overline{x} \to x$. It is folklore and very easy to see that one can characterize the set of satisfying assignments of 2-CNFs in terms of their implication digraphs.

Proposition 4.1.4. *An assignment α satisfies a 2-CNF F if and only if there is no edge in $D(F)$ going out of the set of true literals.*

Assume without loss of generality that F contains no clauses with only one literal, since the value of such literal is forced. We now give a similar characterization of partial prime implicants. For a restriction ρ we partition the set of literals into three sets A_ρ, B_ρ and C_ρ containing false literals, true literals, and those that are free, respectively.

Proposition 4.1.5. *A restriction ρ is a partial prime implicant if and only if*

1. *there is no edge going out of B_ρ*
2. *there is no edge from C_ρ to A_ρ*
3. *C_ρ is a maximal independent set.*

Proof. $\Rightarrow$: Assume that ρ is a prime implicant. Assume for a contradiction that there is an edge going out of B_ρ, namely $u \to v$. If $v \in A_\rho$, by construction of $D(F)$ we have $\overline{u} \vee v$ as a clause in F. But ρ leaves this clause unsatisfied which is a contradiction. If $v \in C_\rho$, since it is left free by ρ we can set it to 0 and we will get a $1 \to 0$ edge, and hence a contradiction with the same argument. If there is an edge from C_ρ to A_ρ, again since we are free to assign values on C_ρ we can get a $1 \to 0$ edge and reach a contradiction. Furthermore, C_ρ is an independent set, since otherwise there would be a clause which is left completely untouched by ρ, and its maximality follows from minimality of ρ.

$\Leftarrow$: We first show that ρ satisfies the formula. Notice that all the clauses are hit by ρ, since C_ρ is an independent set. To see that the formula is indeed satisfied, consider an arbitrary clause $u \vee v$. If both u and v are fixed by ρ, since there is no edge from a true literal to a false literal, the clause should evaluate to true. If u is fixed but v is left free, u has to be set to true, since otherwise there will be an edge from $\overline{u}$ to v contradicting the fact that there

is no edge going out of A_ρ. To show that ρ is a prime implicant, note that if we can unspecify some literal and still satisfy the formula, this new set of free variables should be an independent set, because otherwise some clause will be untouched and hence unsatisfied. □

Claim 5. *For any two distinct partial prime implicants ρ and ρ' we have $C_\rho \neq C_{\rho'}$.*

Proof. For any partial prime implicant ρ since C_ρ is a maximal independent set, we have $A_\rho = N^-(C_\rho)$ and $B_\rho = N^+(C_\rho)$, where for any S, $N^-(S)$ and $N^+(S)$ denote the set of in-neighbors and out-neighbors of S, respectively. Therefore if two partial prime implicants define the same C, they should also define the same A and B, and hence they should be the same. □

We say that an independent set $S \subseteq D(F)$ is *reflexive* if for any literal u, $u \in S$ if and only if $\overline{u} \in S$. By Claim 5, for each partial prime implicant ρ, C_ρ is a maximal reflexive independent set. It is thus enough to bound the number of maximal reflexive independent sets in $D(F)$. From F we construct another graph $G(F)$ on vertices $x_1, \ldots, x_n$ as follows: we include an edge (x_i, x_j) if and only if there is some clause in F which includes literals on both x_i and x_j. We claim that if S is a maximal reflexive independent set in $D(F)$, the set of all variables T appearing in S also forms a maximal independent set in $G(F)$. Assume that there is an edge (x_i, x_j) in T. By construction of $G(F)$ this implies that there is an edge between some literal on x_i and some literal on x_j. But since both literals appear in S, this means that there is an edge in S and hence a contradiction. To see that T is maximal, assume for a contradiction that we can add a variable x_k to T without forming any edge. This means that there is no edge between any literal on any variable in T with any literal on x_k. But this implies that we can add x_k and $\overline{x}_k$ to S and have an independent set, contradicting the maximality of S.

It thus remains to bound the number of maximal independent sets in $G(F)$, which is just a simple undirected graph on n vertices. This is a very well-known problem for which sharp bounds are known.

Theorem 4.1.6 (Moon, Moser [38])**.** *In any simple graph on n vertices, there are at most $3^{\frac{n}{3}}$ maximal independent sets.*

The set of prime implicants consists of those that are partial and those that fix all the variables. Therefore the total of number of prime implicants is bounded by $3^{\frac{n}{3}} + 2^{\frac{n}{2}}$.

4.2 $k \geq 3$[2]

In this section we study the number of prime implicants of k-CNF formulas when $k \geq 3$. We first prove the following lower bound.

Theorem 4.2.1. *For large enough n and k, there exists k-CNF formula F on n variables such that $|\mathsf{PI}(F)| \geq 3^{(1-O(\frac{\log k}{k}))n}$ prime implicants.*

Proof. We follow the construction of Chandra and Markowsky [16]. We divide the set of n variables in n/k parts, each of size k. On each of these parts, we represent the Chandra-Markowsky function asa k-CNF, that is the disjunction of all conjunctions of $2k/3$ variables, exactly $k/3$ of which are negated. Formula F would then be obtained by conjuncting all these function together. In [16] it was shown that each block has at least $\Omega(3^k/k)$ prime implicants. It is easy to see that prime implicants of F are obtained by concatenating prime implicants of the blocks. Therefore the total number of prime implicants is at least $\Omega((3^k/k)^{n/k}) = 3^{(1-O(\log k/k))n}$. □

Generalizing the Satisfiability Coding Lemma [41], we give a scheme to encode prime implicants of k-CNF formulas. Unfortunately our argument goes through as long as the formula is read-$O(1)$.

Theorem 4.2.2. *Let F be an r-read k-CNF on n variables. We have $|\mathsf{PI}(F)| \leq 3^{(1-\frac{1}{kr})n}$.*

Proof. Given a k-CNF F on $x_1, \ldots, x_n$, we construct a CSP F' on $x'_1, \ldots, x'_n$ as follows. We set $\Sigma = \{0, 1, *\}$. Each assignment α on F' naturally induces a restriction σ_α on F as follows: for all $1 \leq i \leq n$, $\sigma_\alpha(x_i) = \alpha(x'_i)$. We want to add constraints to F' so that $F'(\alpha) = 1$ if and only if σ_α is a prime implicant of F. We claim it is possible to define such F' by a $(3, rk)$-CSP. We can define F' as follows

$$F'(\alpha) = 1 \Leftrightarrow F|_{\sigma_\alpha} \equiv 1 \wedge \forall \sigma'(\sigma' \sqsubseteq \sigma_\alpha, |\sigma'| = |\sigma_\alpha| - 1 \Rightarrow F|_{\sigma'} \not\equiv 1).$$

The first conjunct can easily be expressed by a $(3, k)$-CSP. Since

$$F|_{\sigma_\alpha} \equiv 1 \Leftrightarrow \forall C \in F, C|_{\sigma_\alpha} \equiv 1,$$

for each clause $C \in F$ we can introduce a constraint C' in F' such that $C'(\alpha) = 1$ if and only if $C|_{\sigma_\alpha} \equiv 1$. For the second conjunct note that if σ' is obtained from σ_α by unspecifying a variable x_i, $F|_{\sigma'} \not\equiv 1$ can be written as $\bigvee_{C \ni x_i} C|_{\sigma'} \not\equiv 1$. But as x_i appears in at most r clauses each of size at most k, the total number of variables appearing in this expression is at most rk and

[2]This section is based on an unpublished work of Navid Talebanfard

thus we can represent it by a constraint on rk variables. However we do not need this constraint for nx_j which is not fixed by σ_α.

$$D_i := (\alpha(x_i') = *) \vee \bigvee_{C \ni x_i} C|_{\sigma_\alpha^{[x_i \to *]}} \not\equiv 1$$

Thus

$$F'(\alpha) = (\wedge_{C \in F} C') \wedge (\wedge_{i=1}^{n} D_i)$$

Lemma 4.2.3. *All satisfying assignments of F' are isolated.*

Proof. The construction guarantees that $F'(\alpha) = 1$ if and only if σ_α is a prime implicant of F. Let α be such an assignment. We show that if we change the value of any single variable, the restriction induced by the resulting assignment is not a prime implicant. Let x_i' be any variable in F'. Assume $\alpha(x_i') = *$. If we fix x_i' to either 0 or 1, this will correspond to a subcube strictly contained in the one defined by σ_α, and hence contradicting the maximality of σ_α. If on the other hand $\alpha(x_i') = 0$ or 1, then changing this value to $*$ corresponds to inducing a subcube, strictly containing the one defined by α, again contradicting maximality. □

It thus remains to bound the number of isolated solutions of CSPs.

Lemma 4.2.4. *In any (d,k)-CSP on n variables, the number of isolated solutions is at most $3^{(1-\frac{1}{k})n}$.*

Proof. Given F, a (d,k)-CSP on $z_1, \ldots, z_n$ that can take values in Σ, and an isolated assignment α we give an encoding of α with respect a permutation π of the variables.

```
encode(α, π, F)
1: β := the empty string
2: for z_i ∈ vbl(F) in the order given by π do
3:    if F ⊭ z_i = α_i then
4:       append α_i to β
5:    end if
6:    F := F^[z_i ↦ α_i]
7: end for
8: return β
```

Lemma 4.2.5. $\mathbb{E}_\pi[|\mathsf{encode}(\alpha, \pi, F)|] \leq (1 - \frac{1}{k})n$.

Proof. For each z_i, there exists a constraint C_i such that $C_i(\alpha) = 1$, but if we change the value of z_i, C is falsified. Therefore if z_i appears after all other variables in C_i, the encoding procedure is forced to set the value of z_i according to α_i and hence z_i does not appear in the encoding. The expected number of such z_is is at least n/k, and hence the lemma goes through. □

Lemma 4.2.6. *Let $S \subseteq \Sigma^*$ be a prefix-free code with average code length at most ℓ. We have $|S| \leq |\Sigma|^\ell$.*

Proof. Let $d = |\Sigma|$. By Kraft's inequality we have $\sum_{w \in S} d^{-|w|} \leq 1$. On the other hand we have $\ell \geq \sum_{w \in S} \frac{|w|}{|S|}$. We can thus write

$$\begin{aligned}
\ell - \log_d |S| &\geq \sum_{w \in S} \frac{1}{|S|}(|w| - \log_d |S|) \\
&= -\sum_{w \in S} \frac{1}{|S|}(\log_d d^{-|w|} + \log_d |S|) \\
&= -\sum_{w \in S} \frac{1}{|S|} \log_d(|S| d^{-|w|}) \\
&\geq -\log_d(\sum_{w \in S} d^{-|w|}) \\
&\geq 0.
\end{aligned}$$

□

Let S to be the set of isolated solutions of F. Let S_π be the set of encodings of elements of S under π. By Lemma 4.2.5, it follows that there exists some π^* such that the average code length in S_{π^*} is at most $(1 - \frac{1}{k})n$. Then by Lemma 4.2.6 we can bound $|S_{\pi^*}| \leq |\Sigma|^{(1-\frac{1}{k})n}$. But $|S| = |S_{\pi^*}|$ and we are done.

□

All satisfying assignments of F' are isolated and since F' is a $(3, rk)$-CSP, we can use Lemma 4.2.5 and bound the number of prime implicants of F by $3^{(1-\frac{1}{rk})n}$, proving Theorem 4.2.2.

□

Chapter 5

Circuit Complexity of Properties of Graphs with Bounded Planar Cut-width[1]

5.1 Introduction

We consider several of the classical graph decision problems, namely those of deciding existence of 2- and 3-colorings, perfect matchings, Hamiltonian cycles, and disjoint paths. For these problems we are interested in their complexity in the setting of bounded *planar cutwidth*. The *cutwidth* of a graph $G = (V, E)$ with $n = |V|$ vertices is defined in terms of linear arrangements of the vertices. A linear arrangement is simply a 1-1 map $f : V \to \{1, \ldots, n\}$, and its cutwidth is the maximum over i of the number of edges between $V_i = \{v \in V \mid f(v) \leqslant i\}$ and $V \setminus V_i$. The cutwidth of G is the minimum cutwidth of a linear arrangement. Similarly, if the graph G is planar we can define a notion of *planar cutwidth*. Given a linear arrangement f we consider a planar embedding where vertex v is placed at coordinate $(f(v), 0)$. The planar cutwidth of this embedding is then the maximum number of edge-crossings at a vertical line in the plane. We define the planar cutwidth as the minimum planar cutwidth of such a linear arrangement and an embedding.

All the problems we consider can be decided in NC^1 for graphs of bounded cutwidth, and they are in fact NC^1-complete under projection reductions. Imposing planarity, or more precisely considering graphs of bounded planar cutwidth, we are able to place several of the problems in smaller classes such as AC^0, $\mathsf{AC}^0[2]$, and ACC^0, while for some problems they remain NC^1-complete.

Before stating our results we review known complexity results about the graph problems without restriction on cutwidth and the consequences of imposing planarity, for comparison with our results in the bounded cutwidth

[1] This chapter is based on [30]

setting. The 2-coloring problem is in L, as an easy consequence of Reingold's algorithm for undirected connectivity [46], whereas 3-coloring is NP-complete and remains so for planar graphs by the existence of a cross-over gadget [26]. The complexity of deciding if a graph has a perfect matching is still not known. It belongs to P, but it is an open problem if it belongs to NC. For planar graphs the problem is known to be in NC as shown by Vazirani based on work of Kasteleyn [36, 55]; for planar bipartite graphs the problem was shown to be in UL by Datta et al. [20]. The Hamiltonian cycle problem is NP-complete and as shown by Garey et al. it remains so for planar graphs [27], and Itai et al. showed it is NP-hard even for grid graphs [35].

The disjoint paths problem has numerous variations. In the general setting we are given pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$ in a graph G, and are to decide whether disjoint paths between s_i and t_i for each i exists. Here disjoint may mean either vertex-disjoint or edge-disjoint, but either variant is reducible to the other. We shall consider only the case of constant k. When G is an undirected graph a polynomial time algorithm was given by Robertson and Seymour [47], as a result arising from their seminal work on graph minors. When G is a directed graph the problem is NP-complete already for $k = 2$ as shown by Fortune et al. [25]. On the other hand, when G is planar Schrijver [49] gave a polynomial time algorithm for the vertex-disjoint paths problem. The reduction between the vertex-disjoint and the edge-disjoint versions of the problem does not preserve planarity, and it is an open problem the edge-disjoint paths problem in planar directed graphs is NP-complete or solvable in polynomial time [19]. It can however be solved in polynomial time for (not necessarily planar) directed acyclic graphs [25].

Results and techniques

A convenient way to obtain an NC^1 upper bound is through monadic second order (MSO) logic. Elberfeld et al. [23] showed that MSO-definable problems can be decided in NC^1 when restricted to input structures of bounded treewidth, when a tree decomposition of bounded width is supplied in the so-called term representation. We shall not formally define the tree-width of a graph, but we will note that the treewidth of a graph is bounded from above by the cutwidth of the graph [14]. Furthermore, given as input a linear arrangement of bounded cutwidth k, a tree decomposition of tree width k can be constructed by an AC^0 circuit. Thus we have the following meta-theorem as an easy consequence.

Theorem 5.1.1. *Any graph property definable in monadic second order logic with quantification over sets of vertices and edges can be decided by* NC^1 *circuits on graphs of bounded cutwidth if a linear arrangement of bounded cutwidth is supplied as auxiliary input.*

Actually to obtain this we may proceed more directly avoiding the challenges dealt with in [23], since the tree decomposition computed above is actually a path decomposition. We may thus also use standard finite automata rather than tree automata.

All the graph properties we consider can easily be expressed in monadic second order logic, thereby establishing NC^1 upper bounds. We can show that all these problems are in fact also hard for NC^1 under projection reductions. This is based on Barrington's characterization of NC^1 in terms of bounded width permutation branching programs [4].

When considering the graph properties for graphs of bounded planar cutwidth we supply as additional input the corresponding embedding of bounded cutwidth of the graph. But before dealing with this issue, we consider special classes of such graphs where such an embedding is implicit. We consider a grid $\Lambda = \{1, \ldots, l\} \times \{1, \ldots, w\}$ of *width* w and *length* l. A *grid graph* $G = (V, E)$ of width w and length l is a graph where $V \subseteq \Lambda$ and all edges are of Euclidean length 1. We think of the vertices with the same first coordinate to be in the same *layer*. A grid graph with (planar) diagonals allows edges of Euclidean length < 2, but no crossing edges.

We relax these requirements further, defining the class of constant width *grid-layered planar* graphs. A grid-layered planar graph $G = (V, E)$ of width w and length l is a graph with no crossing edges where $V \subseteq \Lambda$ and if two vertices (a, b) and (c, d) are connected by an edge, then $|a - b| \leqslant 1$ and the edge is fully contained in the region $[a - 1, a] \times [1, w]$ or the region $[a, a + 1] \times [1, w]$. If we consider bipartite grid-layered planar graphs we assume that the bipartition is defined by the parities of the sums of coordinates of each vertex. All our lower bounds holds for grid graphs or grid graphs with diagonals, and all our circuit upper bounds hold for grid-planar graphs.

Just as 3-coloring and Hamiltonian cycle remains NP-complete for planar graphs, 3-coloring remains hard for NC^1 on constant width grid graphs with diagonals and Hamiltonian cycle remains hard for NC^1 on constant width grid graphs. We show that 2-coloring on constant width grid-planar graphs is in $\mathsf{AC}^0[2]$. This is complemented by an $\mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AC}^0$ lower bound for grid graphs with diagonals. This lower bound is in some sense not far from the $\mathsf{AC}^0[2]$ upper bound. Namely by the approach of Razborov [45] we have that quasipolynomial size randomized XOR $\circ$ AND is equal to quasipolynomial $\mathsf{AC}^0[2]$. Furthermore Allender and Hertrampf [1] show that in fact quasipolynomial size $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{XOR} \circ \mathsf{AND}$ is equal to quasipolynomial size $\mathsf{AC}^0[2]$.

We show that perfect matching is in ACC^0 for bipartite grid-layered planar graphs, and we have an AC^0 lower bound. For non-bipartite grid-layered planar graphs we have a $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{XOR} \circ \mathsf{AND}$ lower bound. For the disjoint paths problem in constant width grid-layered planar graphs we give AC^0 upper bounds for the following 3 settings: (1) node-disjoint paths in directed graphs. (2) edge-disjoint paths in upward planar graphs. (3) edge-disjoint paths in undirected graphs. We leave open the case of edge-disjoint paths in directed

graphs. For all the settings we have an AC^0 lower bound. All these results are summarized in Figure 5.1.

Problem	Upper bound	Lower bound
2-coloring	$\mathsf{AC}^0[2]$	$\mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AC}^0$
3-coloring	NC^1	NC^1
Bipartite perfect matching	ACC^0	AC^0
Perfect matching	NC^1	$\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{XOR} \circ \mathsf{AC}^0$
Hamiltonian cycle	NC^1	NC^1
Disjoint paths variants	AC^0	AC^0

Figure 5.1: Complexity of problems on constant width grid-layered planar graphs

We shall now discuss extending the upper bounds above from constant width grid-layered planar graphs to the larger classes of graphs of bounded planar cutwidth. Whereas the embedding was implicitly given for grid-layered planar graphs, for graphs of bounded planar cutwidth we will supply a representation of the embedding in addition to the linear arrangement of the vertices. A simple way to represent both the linear arrangement and the planar embedding of a graph $G = (V, E)$ of bounded planar cutwidth is to provide instead a grid-layered planar graph $G' = (V', E')$, where $V \subseteq V'$, where the vertices V are placed on a horizontal line and the vertices $V' \setminus V$ are dummy vertices describing the embedding of the edges. When given this representation as input, our upper bounds are easily adapted. Namely, for the disjoint paths problems an edge can be replaced by a path, and we may simply promote the dummy vertices to regular vertices. For 2-coloring and perfect matching an edge can be replaced by a path of odd length, but this can be done by an AC^0 circuit making locally use of the coordinates of vertices. Namely, we can just ensure that the path alternates between vertices of the implicit bipartition, except possibly at the end.

Our upper bounds are based on reducing to *word problems* on appropriately defined finite monoids. By results of Barrington and Thérien, we then get circuit upper bounds depending on the group structure of the given monoid. The general idea is as follows. For a fixed width w we may view the set of width w grid-layered planar graphs as a free semigroup under concatenation. Consider now a finite monoid $\mathscr{M}$. For each grid-layered planar graph G we associate a monoid element $G^{\mathscr{M}}$. In the simplest setting we will be able to determine if the graph property under consideration holds for the graph G directly from the monoid element $\mathscr{M}$. We will also have defined the elements of $\mathscr{M}$ and the monoid operation in such a way that the map $G \mapsto G^{\mathscr{M}}$ is a homomorphism. What then remains is to analyze the groups inside $\mathscr{M}$. For the disjoint paths problem we show that all groups are trivial, and this gives AC^0 circuits. For 2-coloring we characterize the groups as being isomorphic to

groups of the form $\mathbb{Z}_2^l$, and this gives $\mathsf{AC}^0[2]$ circuits. For perfect matching in bipartite graphs we are not able to fully analyze the groups of the corresponding monoid. We are however able to rule out groups of order 2, and thus by the celebrated Feit-Thompson theorem all remaining groups must be solvable, and this gives ACC^0 circuits.

5.2 Preliminaries

Boolean circuits We give here standard definitions of the Boolean functions and circuit classes we consider. As is usual, when considering a Boolean function $f : \{0,1\}^n \to \{0,1\}$, unless otherwise specified we always have a family of such functions in mind, one for each input length. AC^0 is the class of polynomial size constant depth circuits built from unbounded fanin AND and OR gates. $\mathsf{AC}^0[m]$ allows in addition the function MOD_m given by $\mathrm{MOD}_m(x_1, \ldots, x_k) = 1$ if and only if $\sum_{i=1}^k x_i \not\equiv 0 \pmod m$. We shall also denote the function MOD_2 by XOR. The union of $\mathsf{AC}^0[m]$ for all m is the class ACC^0. NC^1 is the class of polynomial size circuits of depth $O(\log n)$ built from fanin 2 AND and OR gates.

A class of Boolean functions immediately defines a class of Boolean circuits as families of single gate circuits. Given two classes of circuits $\mathcal{C}_1$ and $\mathcal{C}_2$ we denote by $\mathcal{C}_1 \circ \mathcal{C}_2$ the class of circuits consisting of circuits from $\mathcal{C}_1$ that is fed as inputs the output of circuits from $\mathcal{C}_2$. For instance, $\mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AC}^0$ is the class of polynomial size constant depth circuits that has an AND gate at the output, followed by XOR gates that in turn take as inputs the output of AC^0 circuits.

Semigroups, monoids and programs A semigroup is a set $\mathscr{S}$ with an associative binary operation. A monoid $\mathscr{M}$ is a semigroup with a two-sided identity. A subset $\mathcal{G}$ of $\mathscr{M}$ is a group in $\mathscr{M}$ if it is a group with respect to the operation of $\mathscr{M}$. We also say that $\mathscr{M}$ contains $\mathcal{G}$. A monoid is aperiodic if every group it contains is trivial; it is solvable if every group it contains is solvable. A monoid which is not solvable is called unsolvable.

We consider the *program over monoid* [7] formalism for computing Boolean functions. Let $\mathscr{M}$ be a monoid and n an input length. An *instruction* is a triple $\langle j, a_0, a_1 \rangle$, where $j \in [n]$ and $a_0, a_1 \in M$. A program over $\mathscr{M}$ is a pair (P, A) where $A \subset \mathscr{M}$ is the accepting set and $P = (I_1, \ldots, I_\ell)$ is a list of instructions. The length of the program is ℓ. Let $x \in \{0,1\}^n$. The output $I(x)$ of an instruction $I = \langle j, a_0, a_1 \rangle$ is a_{x_j}. The (Boolean) output of the program is is 1 if and only if $\prod_{i=1}^{\ell} I_i(x) \in A$. As with circuits we consider families of programs, one for each input length.

Barrington and Thérien [7] showed that several circuit classes are exactly captured by programs over finite monoids of polynomial length.

Theorem 5.2.1 (Barrington and Thérien)**.** *Let $L \subseteq \{0,1\}^n$.*

- *L is in AC^0 if and only if L is computed by a polynomial length programs over a finite aperiodic monoid.*

- *L is in $\mathsf{AC}^0[m]$ if and only if L is computed by a polynomial length program over a finite solvable monoid in which all groups have orders dividing a power of m*[2]*.*

- *L is in ACC^0 if and only if L is computed by a polynomial length program over a finite solvable monoid.*

- *L is in NC^1 if and only if L is computed by a polynomial length program over a finite unsolvable monoid.*

We shall use only one direction of this characterization, and for this reason it is convenient to reformulate as follows. Let $\mathscr{M}$ be a finite monoid. The *word problem* over $\mathscr{M}$ is to compute the product $x_1 \cdots x_m$ when given as input $x_1, \ldots, x_m \in \mathscr{M}$. When $\mathscr{M}$ is aperiodic the word problem is in AC^0, when $\mathscr{M}$ is solvable and all groups in $\mathscr{M}$ have orders dividing a power of m the word problem is in $\mathsf{AC}^0[m]$, when $\mathscr{M}$ is solvable the word problem is in ACC^0, and we always have the word problem is in NC^1.

5.3 Upper bounds

We first state a geometric lemma that we shall make use of in our results about bipartite matching and disjoint paths. Consider a piecewise smooth infinite simple curve C such that C is contained entirely in the strip $\{(x, y) \mid 1 \leqslant y \leqslant w\}$. We say that C is periodic with period p if the horizontally shifted curve $C + (p, 0)$ coincides with C.

Lemma 5.3.1. *Let $C' = C + (q, 0)$ be a horizontal shift of the curve C. Then C and C' intersect.*

Proof. Map the region $R = \{(x, y) \mid 0 \leqslant x \leqslant p, 1 \leqslant y \leqslant w\}$ into the plane by the map $\varphi(x, y) = (y \cos(2\pi x/p), y \sin(2\pi/p))$. Then $\varphi(C)$ and $\varphi(C')$ are closed simple curves in the plane both containing the origo. By the Jordan curve theorem each of these curves divide the plane into an inside set and an outside set. If they do not intersect then either $\varphi(C)$ encloses $\varphi(C')$ or $\varphi(C')$ encloses all of $\varphi(C)$. In particular this means that the two curves enclose sets of different areas. However $\varphi(C')$ is just a rotation of $\varphi(C)$ around the origo, and must in particular enclose the exact same area as $\varphi(C)$. We conclude the curves intersect. □

[2]This equivalence is not stated explicitly in [7], but follows from the given proof.

2-coloring

We prove here our upper bound for 2-coloring.

Theorem 5.3.2. *Testing whether a given grid-layered planar graph is 2-colorable can be done in* $\mathsf{AC}^0[2]$.

We prove this result by reducing 2-coloring to the word problem a finite monoid $\mathscr{M}$. We then show that all groups in $\mathscr{M}$ are solvable and of order a power of 2. By Theorem 5.2.1 this gives us our $\mathsf{AC}^0[m]$ upper bound.

Reduction to Monoid a Word Problem: A grid-layered planar graph G gives rise to a binary relation $\mathscr{R}(G) \subseteq 2^{\{1,\ldots,w\}} \times 2^{\{1,\ldots,w\}}$. Here $\{1,\ldots,w\}$ are the numbering of the vertices in the layers of the graph. We have that $(S,T) \in \mathscr{R}(G)$ if and only if there is a two-coloring of G such that the vertices in the first layer colored 1 is the set S and the vertices in the last layer colored 1 is the set T. Let $\mathscr{M}$ be the monoid of all such relations under normal composition of relations. Let $G \circ H$ denote the concatenation of the graphs G and H.

Lemma 5.3.3. $\mathscr{R}(G \circ H) = \mathscr{R}(G)\mathscr{R}(H)$

Proof. $\subseteq$: Let $(S,T) \in \mathscr{R}(G \circ H)$. Consider any 2-coloring of $G \circ H$ that witnesses this. Under this coloring all vertices in S in the first layer and all vertices in T in the last layer are assigned color 1. Let U be the set of vertices that get color 1 on the layer that G and H meet. Then obviously $(S,U) \in \mathscr{R}(G)$ and $(U,T) \in \mathscr{R}(H)$ which means $(S,T) \in \mathscr{R}(G)\mathscr{R}(H)$.

$\supseteq$: Let $(S,T) \in \mathscr{R}(G)\mathscr{R}(H)$. Then there exists U such that $(S,U) \in \mathscr{R}(G)$ and $(U,T) \in \mathscr{R}(H)$. Let σ and π be 2-colorings of G and H, respectively, that witness this. Then we can get a 2-coloring of $G \circ H$ by $\sigma\pi$, i.e., by coloring G using σ and then extending it via π on H. □

The proof of the upper bound is now completed by the following result.

Proposition 5.3.4. *Every group* $\mathscr{G} \subseteq \mathscr{M}$ *is isomorphic to* $\mathbb{Z}_2^\ell$ *for some* ℓ.

Proof. For a graph G, let us identify the nodes in the first layer with the set $\{1,\ldots,w\}$ and the nodes in the last layer with the set $\{1',\ldots,w'\}$.

The observation that makes the proof possible is the following: Suppose $\mathscr{R}(G) = \mathscr{R}(H)$. Then for any $u,v \in \{1,\ldots,w\} \cup \{1',\ldots,w'\}$ we have that u and v are connected in G if and only if u and v are connected in H. Furthermore if u and v are connected in G by an odd (even) length path then u and v are connected in H by an odd (even) length path.

Suppose now that G is two-colorable. Then every connected component of G can be two-colored in exactly two different ways. This means that $\mathscr{R}(G)$ can be reconstructed from only the following information about G: Which nodes

from $\{1,\ldots,w\}\cup\{1',\ldots,w'\}$ are connected in addition to a single two-coloring of those vertices.

Let $\mathscr{G} \subseteq \mathscr{M}$ be a group with identity E. For any element A in $\mathscr{G}$ fix a grid-layered planar graph $G(A)$ such that $\mathscr{R}(G(A)) = A$.

Let $(i \sim j) \in G$ denote that i and j are connected via a path in G.

Claim 6. *Let $A, B \in \mathscr{G}$. Then $(i \sim j) \in G(A)$ if and only if $(i \sim j) \in G(B)$, and $(i' \sim j') \in G(A)$ if and only if $(i' \sim j') \in G(B)$.*

Proof. It is enough to prove the claim for the case when B is just the identity element E. Assume $(i \sim j) \notin G(A)$. Since $EA = A$ and that $\mathscr{R}(G(E) \circ G(A)) = EA = A$ by Lemma 5.3.3, there cannot be a path between i and j in $G(E) \circ G(A)$ since otherwise the color of i and j will depend on each other and we know that this is not the case since $(i \sim j) \notin G(A)$. Therefore $(i \sim j) \notin G(E) \circ G(A)$ which in particular implies $(i \sim j) \notin G(E)$. For the other direction assume that $(i \sim j) \notin G(E)$. We have $AA^{-1} = E$. Using Lemma 5.3.3 we get $\mathscr{R}(G(A) \circ G(A^{-1})) = AA^{-1} = E$. This implies that there is no path between i and j in $G(A) \circ G(A^{-1})$ since otherwise the colors of i and j will depend on each other in $G(E)$ which we know is not the case. Therefore $(i, j) \notin G(A) \circ G(A^{-1})$ and hence $(i, j) \notin G(A)$. This shows that $(i, j) \in G(E)$ if and only if $(i, j) \in E(A)$. To show that $(i' \sim j') \in G(A)$ if and only if $(i' \sim j') \in G(E)$ we consider equations $AE = A$ and $A^{-1}A = E$ and use a similar argument as above. □

Let $A \in \mathscr{G}$ and consider the graph $G(A)$. For any set $S \subseteq G(A)$ let $\overleftarrow{V}(S) = S \cap \{1,\ldots,w\}$ and $\overrightarrow{V}(S) = S \cap \{1',\ldots,w'\}$. We define $L(A)$ to be the set of all connected components C in $G(A)$ such that $\overleftarrow{V}(C) \neq \emptyset$ and $\overrightarrow{V}(C) = \emptyset$. Similarly let $R(A)$ denote the set of all connected components C such that $\overleftarrow{V}(C) = \emptyset$ and $\overrightarrow{V}(C) \neq \emptyset$. We now let $V_L(A) = \{\overleftarrow{V}(C) : C \in L(A)\}$ and $V_R(A) = \{\overrightarrow{V}(C) : C \in R(A)\}$. Define $M(A)$ to be the set of connected components that are neither in $L(A)$ nor in $R(A)$ and have vertices on both sides of $G(A)$. We then define $V_L^M(A) = \{\overleftarrow{V}(C) : C \in M(A)\}$ and $V_R^M(A) = \{\overrightarrow{V}(C) : C \in M(A)\}$.

Claim 7. *The following properties hold.*

(i) *$V_L(A) = V_L(E)$ and $V_R(A) = V_R(E)$. Furthermore, for any pair of i and j that are in the same component in $L(A)$, the lengths of all paths between i and j in $G(A)$ have the same parity and that is the same as in $G(E)$. Similarly for every pair i' and j' that are in the same connected component in $R(A)$, the length of all paths between i' and j' are of the same parity and that is the same as in $G(E)$.*

(ii) *$V_L^M(A) = V_L^M(E)$ and $V_R^M(A) = V_R^M(E)$.*

Proof. All these follow straightforwardly from $A^{-1}A = AA^{-1} = E$ and $EA = AE = A$.

(i) Consider a node i in $G(E)$ which appears in some component in $L(E)$. Note that $AA^{-1} = E$. Since the color of i in $G(E)$ does not depend on any node on the right side of $G(E)$ the same should hold for $G(A) \circ G(A^{-1})$. By Claim 6 we know that for any j we have $(i \sim j) \in G(A)$ if and only if $(i \sim j) \in G(E)$. Therefore $V_L(E) = V_L(A)$. Considering $A^{-1}A = E$ and using a similar argument we can show that $V_R(E) = V_R(A)$. The parity of path lengths are preserved because since $G(A) \circ G(A^{-1})$ and $G(E)$ admit the same 2-colorings.

(ii) Let i and j be nodes in some set $S \in V_L^M(A)$. Consider $H = G(E) \circ G(A)$. Then by Claim 6 we know that i and j are connected $G(E)$. But they should also be connected to some node on the right end layer of H, since H and $G(A)$ admit the same colorings on boundaries. This means that i and j are both in some set $T \in V_L^M(E)$. Conversely let i and j be nodes in some set $S \in V_L^M(E)$. This time we let $H = G(A) \circ G(A^{-1})$. Again by Claim 6, i and j are connected in $G(A)$, but they are also connected to some node on the right end layer of $G(A^{-1})$. This means that they are also connected to some nodes on the right end layer of $G(A)$ and thus they belong to some set $T \in V_L^M(A)$. This shows that $V_L^M(A) = V_L^M(E)$. We can prove $V_R^M(A) = V_R^M(E)$ similarly.

□

For each component in $M(A)$ we pick two representatives, one from each side. We pick the left representatives $i_1 < \ldots < i_m$ arbitrarily. But for the right representatives if i_k is connected to i'_k then we pick i'_k as the representative of the k'th component, otherwise we pick an arbitrary node in the component. Let the right representatives be $j'_1 < \ldots < j'_m$. We map the left representative of a component to its right representative. Since $G(A)$ is planar, for every k we have that i_k is mapped to j'_k. This means that we can rename the components in $M(A)$ by $C_1, \ldots, C_m$ such that all vertices in C_i appear after all vertices in C_{i-1}.

Furthermore we know by above claim that in a group, these components are the same on the boundaries of the graph of each group element. For any $A \in \mathscr{G}$ and any $1 \leq k \leq m$ let π_k^A be the parity of the length of all paths between i_k and j'_k in $G(A)$. We show that there exists a sequence $\epsilon_1, \ldots, \epsilon_m \in \{0,1\}^m$ such that for any $A, B \in \mathscr{G}$ and all $1 \leq k \leq m$, the parity of the paths between i_k and j'_k in $G(A) \circ G(B)$ is given by $\pi_k^A \oplus \pi_k^B \oplus \epsilon_k$. To see this consider the graph $G(A) \circ G(B)$ and rename the i_k and j'_k on the side where $G(A)$ and $G(B)$ meet as $i^{(1)}$ and $i^{(2)}$ ($i^{(1)} = i_k$ and $i^{(2)} = j'_k$). If $j'_k = i'_k$ we set $\epsilon_k = 0$. This clearly satisfies the desired property, since to get from i_k on the left layer of $G(A)$ to i'_k on the right layer of $G(B)$ we can first go

to i'_k on the right layer of $G(A)$ and then to i'_k on the right layer of $G(A)$. Any such path has clearly parity $\pi^A_k + \pi^B_k$. If $j'_k \neq i'_k$ we note that the parity between $i^{(1)}$ and $i^{(2)}$ is exactly the same as in $G(E) \circ G(E)$ by Claim 7. We denote this by ϵ_j. Now to color $G(A) \circ G(B)$ if we use color 0 on i_j then we are forced to use color π^A_j on $i^{(2)}$, and hence $\pi^A_j \oplus \epsilon_j$ on $i^{(1)}$ and finally we should use $\pi^A_j \oplus \epsilon_j \oplus \pi^B_j$ on i'_j. This means that the parity between i_j and i'_j is $\pi^A_j \oplus \pi^B_j \oplus \epsilon_j$ as claimed.

We define a group $\mathbb{Z}_2^{(\epsilon_1,\ldots,\epsilon_m)}$ as follows. The elements are just the same as $\mathbb{Z}_2^m$, and the group operation is defined as $\mathbb{Z}_2^m$ but then adding the vector $(\epsilon_1, \ldots, \epsilon_m)$ to the result. It is clear that $\mathbb{Z}_2^{(\epsilon_1,\ldots,\epsilon_m)}$ and $\mathbb{Z}_2^m$ are isomorphic. The above argument shows that $\mathscr{G}$ is isomorphic to $\mathbb{Z}_2^{(\epsilon_1,\ldots,\epsilon_m)}$ and hence to $\mathbb{Z}_2^m$ □

Bipartite matching

We prove here our upper bound for bipartite matching.

Theorem 5.3.5. *Given a bipartite grid-layered planar graph G, we can decide whether G has a perfect matching in* ACC^0.

Reduction to a Monoid Word Problem For each grid-layered planar graph G of odd length ℓ that has no vertical edges in the rightmost layer, we define the corresponding monoid element $G^{\mathscr{M}}$ as the triple (X, Y, R) where $X \subseteq [w]$ is the set of vertices in the leftmost layer of G, $Y \subseteq [w]$ is the set of vertices in the rightmost layer of G and $R \subseteq 2^X \times 2^Y$ is a binary relation such that for any $\mathsf{X}_1 \subseteq X$, $\mathsf{X}_2 \subseteq Y$ we have $(\mathsf{X}_1, \mathsf{X}_2) \in R$ if and only if G has a matching that matches all vertices in G except $\overline{\mathsf{X}_1}$ in the leftmost layer and X_2 in the rightmost layer. The monoid product is defined as $(\mathsf{X}_1, \mathsf{X}_2, R)(\mathsf{X}_3, \mathsf{X}_4, S) = (\mathsf{X}_1, \mathsf{X}_4, R \circ S)$ when $\mathsf{X}_2 = \mathsf{X}_3$ and $\circ$ is the usual composition of binary relations. When $\mathsf{X}_2 \neq \mathsf{X}_3$, we define the product to be an element 0 for which $0x = x0 = 0$ for any x in the monoid. Now define the monoid $\mathscr{M} = \{G^{\mathscr{M}} : G \text{ is an odd length bipartite grid-layered planar graph}\} \cup \{0\} \cup \{1\}$, where 1 is an added identity. It is easy to see that the monoid operation described corresponds to concatenation of graphs (by merging the vertices in the rightmost layer of first graph with the vertices in the leftmost layer of the second graph). So a perfect matching exists in G if and only if $G^{\mathscr{M}} = (X_1, X_2, R)$ and R contains the element $(X_1, \overline{X_2})$.

To show that $\mathscr{M}$ is solvable we will prove that it does not contain any group of order two. We then use Proposition 5.3.6 and Theorem 5.3.7 below to conclude that $\mathscr{M}$ is solvable.

Proposition 5.3.6. *If $\mathcal{G}$ is a finite group of order $2k$ for some $k \geqslant 1$, then there exists $a \in \mathcal{G}$ such that $a \neq e$ and $a^2 = e$, where e is the identity of $\mathcal{G}$.*

Proof. Let $a_1, \ldots, a_{2k-1}$ be the non-identity elements in $\mathcal{G}$. Pair each a_i with its inverse a_j. There will be at least one a_i such that $a_i = a_i{}^{-1}$. So ${a_i}^2 = e$ where e is the identity element in $\mathcal{G}$. □

Theorem 5.3.7 (Feit-Thompson [24])**.** *Every group of odd order is solvable.*

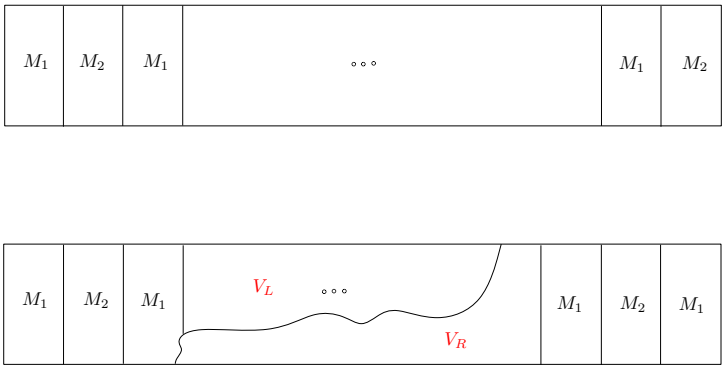
Proposition 5.3.8. *The monoid $\mathscr{M}$ is solvable.*

Proof. We begin by considering an arbitrary group $\mathcal{G} \subset \mathscr{M}$, such that $\mathcal{G} \neq \{0\}$ and $\mathcal{G} \neq \{1\}$. First, observe that for any two elements (X_1, X_2, R) and (X_3, X_4, S) in the group $X_1 = X_2 = X_3 = X_4$ as $0 \notin \mathcal{G}$. So we can identify any element (X_1, X_2, R) of the group by simply using R. Suppose now that $\mathcal{G} = \{E, R\}$ is of order 2, where E is the identity element in $\mathcal{G}$. We will show that $E \subseteq R$. This then means that $R = ER \subseteq R^2 = E$, contradicting the existence of $\mathcal{G}$. Let $k = 2^w$. Suppose $(\mathsf{Y_0}, \mathsf{Y_{k+1}}) \in E$. Since $E^{k+1} = E$ there exists $\mathsf{Y_1}, \ldots, \mathsf{Y_k}$ such that $(\mathsf{Y_i}, \mathsf{Y_{i+1}}) \in E$ for all i. Thus there must exist some $\mathsf{X_1} = \mathsf{Y_i}$ such that $(\mathsf{Y_0}, \mathsf{X_1}) \in E$, $(\mathsf{X_1}, \mathsf{X_1}) \in E$, and $(\mathsf{X_1}, \mathsf{Y_{k+1}}) \in E$. We shall show that $(\mathsf{X_1}, \mathsf{X_1}) \in R$, and since $R = ERE$ this shows also $(\mathsf{Y_0}, \mathsf{Y_{k+1}}) \in R$.

Since $(\mathsf{X_1}, \mathsf{X_1}) \in E$ and $R^2 = E$ there exists $\mathsf{X_2}$ such that $(\mathsf{X_1}, \mathsf{X_2}) \in R$ and $(\mathsf{X_2}, \mathsf{X_1}) \in R$. Consider a graph G defining R, and let M_1 be a matching in G corresponding to $(\mathsf{X_1}, \mathsf{X_2}) \in R$ and let M_2 be a matching in G corresponding to $(\mathsf{X_2}, \mathsf{X_1}) \in R$. Consider now the graph $S = M_1 \cup M_2$. The graph S^n is obtained by concatenating n copies of S. We note that for any n, the graph S^n is a union of two matchings. The matching M obtained by the concatenation of matchings $M_1 M_2 \ldots$ and the matching N obtained by the concatenation of matchings $M_2 M_1 \ldots$.

We label the vertices on the left side on the i^{th} copy of S as $1_{(i)}, \ldots, k_{(i)}$. The rightmost vertices in S^n are labelled $1_{(n+1)}, \ldots, k_{(n+1)}$. A path in S^n is called a *blocking path* if it connects some vertex in the leftmost layer to some vertex in the rightmost layer.

We will show below that for even $n \geq w$ the graph S^n does not have a blocking path, but first we show that this will complete the proof. Suppose that S^n in fact does not have a blocking path. Consider the set V_L of all vertices in S^n that are reachable from some vertex in the left end and the set V_R of all vertices in S^n that are reachable from some vertex in the right end. Put any remaining vertices in the set V_L. Since there is no blocking path V_L and V_R are disjoint. Now we can obtain a matching corresponding to $(\mathsf{X_1}, \mathsf{X_2})$ in $R^n = E$ by using matching M on the vertices in V_L and using matching N on V_R. Since $(\mathsf{X_2}, \mathsf{X_1}) \in R$ this means $(\mathsf{X_1}, \mathsf{X_1}) \in R$ as should be shown.



We say that a path P crosses a boundary in S^n if it has two consecutive edges e_1 and e_2 such that they belong to different copies of S in S^n. Note that e_1 and e_2 must belong to the same matching M_1 or M_2. If they do not, the vertex common to those edges must be in $\mathsf{X}_1 \cap \overline{\mathsf{X}_1}$ or $\mathsf{X}_2 \cap \overline{\mathsf{X}_2}$.

Claim 8. *For any n, the graph S^n cannot have a path from $v_{(i)}$ to $v_{(i+1)}$ for any i and v.*

Proof. Given a graph G, the operation of attaching length 2 horizontal paths to the vertices in the left and right side through two new layers, gives the same monoid element. This is because the vertices in the graph corresponding to the monoid element which were originally matched inside the graph remain matched inside the graph itself and vice versa. Therefore without loss of generality we can assume that the graph we consider has this form.

Suppose such a path P from $v_{(i)}$ to $v_{(i+1)}$ exists. Suppose also that P connects to both these vertices from the same side, left or right. Consider the shifted version P' of P in S^{n+1} from $v_{(i+1)}$ to $v_{(i+2)}$. These paths thus share an edge, but they must diverge at some vertex. This means there exist a vertex of degree at least 3 in S^{n+1} which is impossible since S^{n+1} is a union of two matchings. Thus P must connect to the two vertices $v_{(i)}$ to $v_{(i+1)}$ from opposite sides. This means that it crosses boundaries an even number of times. By bipartiteness the path is of even length, and together this means that the first and last edge can not be from the same matching. This implies that $v \in \mathsf{X}_1 \cap \overline{\mathsf{X}_1}$ or $v \in \mathsf{X}_2 \cap \overline{\mathsf{X}_2}$, contradicting the existence of P.

□

The following last claim completes the proof that $\mathscr{M}$ is solvable.

Claim 9. *S^n does not have a blocking path for $n \geqslant w$.*

Proof. Assume that S^n has a blocking path for $n \geqslant w$. This blocking path must pass each of the $n+1$ boundaries (including left and right ends) at least once. Therefore we can find integers i and j such that this blocking path has a segment P connecting $v_{(i)}$ to $v_{(j)}$ for some $1 \leqslant v \leqslant w$. By Claim 8, we have $j > i+1$. Now consider the graph S^{n+1}. This graph also has this path

P from $v_{(i)}$ to $v_{(j)}$ and also a path P' from $v_{(i+1)}$ to $v_{(j+1)}$ that is simply a "shifted" version of P. By Claim 8, these paths are vertex disjoint. Because if they intersect then we can construct a path from $v_{(i)}$ to $v_{(i+1)}$ in S^{n+1}. By using Lemma 5.3.1 we conclude that the paths P and P' must intersect. This concludes the proof. □

□

Disjoint paths

We consider several different variants of the disjoint paths problem, but there is significant overlap in the different approaches. In each case we define a monoid $\mathscr{M}$ and show it is aperiodic. We can thus compute the word problem over $\mathscr{M}$ by AC^0 circuits and we can use these to solve the disjoint paths problem.

The monoids. We describe here the monoid in general terms. Elements of $\mathscr{M}$ consist of a (downward closed) set of edges between the set of vertices $W = \{1, \ldots, w\} \cup \{1', \ldots, w'\}$. Consider a grid-layered planar graph G. This may be either undirected or directed. We construct a monoid-element $G^{\mathscr{M}}$ from G as follows, by letting every set of disjoint paths in G between vertices from W give rise to a set of corresponding edges in $G^{\mathscr{M}}$. Depending on the setting these paths may be vertex-disjoint or edge-disjoint, and if the graph is directed the edges are directed accordingly. The operation of the monoid will be the natural operation that makes the map $G \mapsto G^{\mathscr{M}}$ a homomorphism. Note that if $A \subseteq A'$ and $B \subseteq B'$ then $AB \subseteq A'B'$.

Reduction to monoid product. Let G be a grid-layered planar directed graph with pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k)$. Consider the partition of G into at most $2k + 1$ segments obtained by dividing at every layer containing a terminal. For each segment we divide the graph into segments of length 1, translate these to monoid elements and compute the product of these. This results in at most $2k+1$ monoid elements describing all possible disjoint paths connecting endpoints of every segment. Since k is fixed this is a fixed amount of information from which it can then be directly decided whether disjoint paths exist between all pairs of terminals.

Showing aperiodicity of the monoid. The approach we will use in all cases is as follows. Let $\mathscr{G}$ be a group in $\mathscr{M}$ with identity E, and let A be any element of $\mathscr{G}$. We shall then prove that $E \subseteq A$. Note then that this means $A^{-1} = EA^{-1} \subseteq AA^{-1} = E$, and hence $A = E$. Showing this for all A implies that $\mathscr{G}$ is trivial.

Edge-disjoint paths in upward planar graphs

Here we consider *directed upward grid-layered planar graphs*, i.e., every edge in the graph is directed from some vertex in layer i to some vertex in layer $i+1$ or it is directed from a layer i vertex to another layer i vertex that is at a distance of one unit on the grid from the source vertex and edges do not cross. We are to decide if edge-disjoint paths exists from $(s_1, t_1), \ldots, (s_k, t_k)$ for fixed k. Thus if we consider the monoid element corresponding to such a graph, each set of edges in the monoid element contains only directed edges from the left-side vertices $\{1, \ldots, w\}$ to the right-side vertices $\{1', \ldots, w'\}$, and these correspond to pairwise edge-disjoint paths in the corresponding directed upward planar grid graph.

Our main theorem in this section is the following.

Theorem 5.3.9. *For any fixed k, given a directed upward grid-layered planar graph G and k pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, we can decide whether there are pairwise edge-disjoint paths from s_i to t_i for $i = 1$ to k in AC^0.*

The theorem follows from the following claim

Claim 10. *$\mathscr{M}$ is aperiodic.*

Proof. Let $\mathcal{G}$ be a group in $\mathscr{M}$ with identity E. Let A be an element of $\mathcal{G}$ such that $o_{\mathcal{G}}(A) = p \geqslant 2$. We are to show that $E \subseteq A$.

Let $G(E)$ and $G(A)$ be grid-layered planar graphs such that $G(E)^{\mathscr{M}} = E$ and $G(A)^{\mathscr{M}} = A$. Let $S \in E$, $c = |S|$ be the number of edges in S between the sets $\{1, \ldots, w\}$ and $\{1', \ldots, w'\}$, and let $t = w^c + 1$. Let these edges be $(s_1, t_1), \ldots, (s_c, t_c)$. Consider the concatenation $G(E)^{t+1}$ of the graph $G(E)$ with itself. Since $E^{t+1} = E$ we have in $G(E)^{t+1}$ disjoint paths corresponding to S, and we will think of S as those paths. Since we have t boundaries between the $t+1$ copies of $G(E)$, there must be two boundaries such that for each of the c paths, the vertices of the two boundaries that the path crosses are the same. Let $j_1 \geqslant j_2 \geqslant \ldots \geqslant j_c$ be these vertex numbers. Splitting the graph $G(E)^{t+1}$ at these two layers divides each path into 3 parts. Thus the middle part consists of edge-disjoint (j_i, j_i)-paths for $i = 1, \ldots, c$. The left part will contain edge-disjoint paths (s_i, j_i) and the right part will contain edge-disjoint (j_i, t_i) paths . Also note that the left part and the right part are also graphs that correspond to the monoid element E; So E must have these paths as well. We will show that $G(A)^{2pc+1}$ also contains such paths. Since $A = EA^{2pc+1}E$ we can concatenate (s_i, j_i) paths from the $G(E)$ on the left side with the concatenation of (j_i, j_i) paths in $G(A^{2pc+1})$ and (j_i, t_i) paths in the $G(E)$ on the right side to show that $S \in A$.

Since $A^p = E$ we have that $G(A)^p$ contains (j_i, j_i)-paths for $i = 1, \ldots, c$. Denote these by $P_1, \ldots, P_c$. These paths are edge-disjoint, but may cross each other at a vertex. We first argue that without loss of generality we may assume that they never cross; they may touch each other at vertices, however.

Indeed, if this is not the case we can construct such paths $\widehat{P_1}, \ldots, \widehat{P_c}$ as follows. Let H be the union of $P_1, \ldots, P_c$. Suppose we have already found the paths $\widehat{P_1}, \ldots, \widehat{P_{i-1}}$. Then let $\widehat{P_i}$ be the top-most (j_i, j_i) path in H, and then erase the edges of $\widehat{P_i}$ from H.

We can think of the paths $P_1, \ldots, P_c$ as infinite paths with period p in an infinite concatenation of the graph $G(A)$. Let $P'_1, \ldots, P'_c$ be the paths obtained by shifting the paths by the length of one graph $G(A)$ to the right. By Lemma 5.3.1 we have that P_i and P'_i must intersect in $G(A)^p$. Let $Q_1, \ldots, Q_c$ be the paths such that Q_i is the upper envelope path of P_i and P'_i. We are now ready to construct the edge-disjoint (j_i, j_i) paths $R_1, \ldots, R_c$ in $G(A)^{2pc+1}$. We think of the graph $G(A)^{2pc+1}$ as $2c$ blocks of $G(A)^p$ followed by a single $G(A)$. The path R_i proceeds as follows. In the first $i-1$ blocks it follows P_i. Then in block i, when P_i intersects Q_i it follows Q_i, and continues to do so for the following $2(c-i)$ blocks. Then in block $2c-i+1$ when P'_i intersects Q_i it follows P'_i, and continues to do so for the remaining $i-1$ blocks. After these $2c$ blocks, the c paths that started out as $P_1, \ldots, P_c$ are ending as $P'_1, \ldots, P'_c$ and they follow $P'_1, \ldots, P'_c$ through the last graph $G(A)$, thereby making Q_i a (j_i, j_i)-path. We claim that the paths $R_1, \ldots, R_c$ are edge-disjoint. Observe the path R_i. In the first $i-1$ blocks it cannot intersect any of the paths $R_1, \ldots, R_{i-1}$. This is because $R_i = P_i$ for these blocks and R_j for $j < i$ is either P_j in which case R_i is disjoint from these as P_i is disjoint from P_j for all j or R_j is the upper envelope of P_j and P'_j in which case R_j can only move further away from R_i. After that all R_is start following Q_is and they remain edge-disjoint as Q_is are edge-disjoint. Now note that R_i switches to P'_i before R_j for $j < i$. So R_i cannot intersect with R_j as R_i can only move further away from R_j (which is still following the upper envelope). Now once all R_is have switched to P'_is they remain edge-disjoint as P'_is are edge-disjoint.

□

Vertex-disjoint paths

Our main theorem in this section is the following.

Theorem 5.3.10. *For any fixed k, given a directed grid-layered planar graph G and k pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$, we can decide whether there are pairwise vertex-disjoint paths from s_i to t_i for $i = 1$ to k in AC^0.*

A monoid element $G^{\mathscr{M}}$ consists of sets of directed edges on the set $\{1, \ldots, w\} \cup \{1', \ldots, w'\}$ that form partial matchings, that is no two edges share an endpoint. Note that the edges could go from a vertex to a vertex on the same side (For ex., from 1 to 2 or from $1'$ to $2'$).

The theorem follows from the following claim.

Claim 11. *$\mathscr{M}$ is aperiodic.*

Proof. Let $\mathcal{G}$ be a group in $\mathscr{M}$ with identity E. Let A be an element of $\mathcal{G}$ such that $o_{\mathcal{G}}(A) = p \geqslant 2$. We are going to show that $E \subseteq A$.

Let $G(E)$ and $G(A)$ be grid-layered planar graphs such that $G(E)^{\mathscr{M}} = E$ and $G(A)^{\mathscr{M}} = A$. Let $S \in E$ and let $t = w^{w+1} + 1$. We shall prove that $S \in A$. Let c be the number of edges in S between the sets $\{1, \ldots, w\}$ and $\{1', \ldots, w'\}$. We call the corresponding paths crossing paths.

Consider the concatenation $G(E)^{t+1}$ of the graph $G(E)$ with itself. Since $E^{t+1} = E$ we have in $G(E)^{t+1}$ disjoint paths corresponding to S. We will think of S also as this set of paths. This set will induce vertex disjoint paths in each of the $t+1$ copies of $G(E)$. We will now have two cases: Either all the $t+1$ graphs have exactly c crossing paths or some graph has $c' > c$ crossing paths. In case some graph $G(E)$ has $c' > c$ crossing paths, let S' be the set of paths induced by S in that graph. We then start over, and prove that $S' \in A$. This will imply that $S \in A$ since $A = EAE$, and this case can only occur a finite number of times since $c' \leqslant w$.

So we may now suppose that all graphs have exactly c crossing paths. This means also that each path in S crosses each graph $G(E)$ of the concatenation $G(E)^{t+1}$ using a single crossing path. Between two such crossing paths, a path (in S) may cross the boundary between two graphs a number of times. Record for each boundary and for each path an ordered list of the vertex numbers in which the path crosses the boundary. By the choice of t there must be two boundaries for which each path cross in the same way in the two boundaries. We then consider the part of the concatenation between these two boundaries and let S' be the disjoint paths induced on this part. Clearly $S' \in E$ as we have taken S' from a concatenation of $G(E)$ graphs, and we will prove that $S' \in A$ and be done again since $A = EAE$.

We have a set of disjoint paths corresponding to S' in the graph $G(A)^p$ (Since $A^p = E$). We can think of these paths as being induced by infinite paths $P_1, \ldots, P_c$ with period p in an infinite concatenation of the graph $G(A)$. We suppose these are ordered such that P_1 is the top-most path, P_2 the second top-most path, and so on. Let $P_1', \ldots, P_c'$ be the infinite paths obtained by shifting the paths by the length of one graph $G(A)$ to the right. By Lemma 5.3.1 we have that P_i and P_i' must intersect in $G(A)^p$. Let $Q_1, \ldots, Q_c$ be the paths such that Q_i is the upper envelope path of P_i and P_i'.

We are now ready to construct vertex-disjoint paths $R_1, \ldots, R_c$ corresponding to S in $G(A)^{(2c+2)p+1}$. As before we consider $G(A)^{(2c+2)p+1}$ as $2c+2$ blocks of $G(A)^p$ followed by a single $G(A)$. Path R_i proceeds as follows. In the first block it follows exactly along P_i, and it continues to do so for the next $i-1$ blocks. Then in block $i+1$, when P_i intersects Q_i it follows Q_i. Note that this may lead R_i into the previous block, but it will not intersect itself. From block $i+1$ the path R_i continues to follow along Q_i for the following $2(c-i)$ blocks. Then in block $2c-i+2$ when P_i' intersects Q_i it follows P_i' and continues to do so for the next $i-1$ blocks. After these $2c+1$ blocks the paths R_i continue to follow along P_i' for another block, and also through the

last graph $G(A)$. Using an argument similar to the one we used in proving Claim 10, we can conclude that $R_1, \ldots, R_c$ are pairwise vertex-disjoint and this completes the proof. □

Edge-disjoint paths in undirected graphs

Here we consider the setting where the graphs are undirected, and we are to decide if edge-disjoint paths exists. Thus the monoid elements consists of sets of undirected edges. The proof will use ideas from both of the two previous paragraphs.

Aperiodicity of the monoid. Let $\mathscr{G}$ be a group in $\mathscr{M}$ with identity E. Let A be an element of $\mathscr{G}$ of period p. We are to show that $E \subseteq A$.

The proof first proceeds exactly as in the vertex-disjoint case. Thus we let $G(E)$ and $G(A)$ be grid-layered planar graphs such that $G(E)^{\mathscr{M}} = E$ and $G(A)^{\mathscr{M}} = A$. Let $S \subseteq E$ and let c be the number of cross-edges in S. By considering the concatenation $G(E)^{t+1}$ with $t = w^{w+1} + 1$, we may reduce to the case where we have a set of disjoint paths corresponding to S in the graph $G(A)^p$, and where we can this of these paths as being induced by infinite paths $P_1, \ldots, P_c$ with period p in an infinite concatenation of the graph $G(A)$.

Now, we depart from the similarity with the vertex-disjoint case. The infinite paths $P_1, \ldots, P_c$ are just assumed to be edge-disjoint so they may intersect at vertices. From these we shall now construct infinite paths $\widehat{P_1}, \ldots, \widehat{P_c}$ that are edge disjoint, but may only touch at vertices, never cross. From the construction the paths are ordered from top path $\widehat{P_1}$ to the bottom path $\widehat{P_c}$. Let H be the union of all the paths infinite paths $P_1, \ldots, P_c$. Suppose we have already found the paths $\widehat{P_1}, \ldots, \widehat{P_{i-1}}$. Let $\widehat{P_i}$ be the upper-envelope path in H, and then erase the edges of $\widehat{P_i}$ from H. (This is the step that does not generalize to the case directed graphs). Assume from now on that the original paths $P_1, \ldots, P_c$ are ordered in such a way that P_i intersects $\widehat{P_i}$.

Let $P_1', \ldots, P_c'$ be the infinite paths obtained by shifting the paths $P_1, \ldots, P_c$ by the length of one graph $G(A)$ to the right. Similarly, let $\widehat{P_1}', \ldots, \widehat{P_c}'$ be the infinite paths obtained by shifting the paths $\widehat{P_1}, \ldots, \widehat{P_c}$ by the length of one graph $G(A)$ to the right. Let $Q_1, \ldots, Q_c$ be the paths such that Q_i is the upper envelope path of $\widehat{P_i}$ and $\widehat{P_i}'$.

We are now ready to construct new paths $R_1, \ldots, R_c$. These are constructed in blocks of $G(A)^p$, and follows several phases. We start out with the paths $P_1, \ldots, P_c$. These are followed for one block. Then we have a transition phase to the paths $\widehat{P_1}, \ldots, \widehat{P_c}$. This gives the geometric ordering and in the next two transition phases we transition to the shifted versions $\widehat{P_1}', \ldots, \widehat{P_c}'$ via the upper envelope paths $Q_1, \ldots, Q_c$. Finally in the last transition phase we transition back to the original (but shifted) paths $P_1', \ldots, P_c'$. These are then

followed through one block and then one more graph $G(A)$ to complete the paths.

We shall describe the transition between the paths $P_1, \ldots, P_c$ and $\widehat{P_1}, \ldots, \widehat{P_c}$, and the rest of the proof then follows analogously to the previous settings. Here we are in the situation that the paths $R_1, \ldots, R_c$ have followed the paths $P_1, \ldots, P_c$ for one block. Then in the next block, when P_1 first intersects with $\widehat{P_1}$ we will let R_1 follow along $\widehat{P_1}$, and we do this by also modifying the other paths $R_2, \ldots, R_c$ accordingly. More precisely, whenever two or more paths meet at a later vertex of $\widehat{P_1}$ we exchange the continuations of curves if necessary in order to let R_1 follow along $\widehat{P_1}$. We continue in a similar way in the next $c-1$ blocks, letting R_j start following along $\widehat{P_j}$ beginning from block j.

Performing all the transitions as described above we have shown that $S \in A^{p(1+4c)+1} = A$, thereby completing the proof.

5.4 Hardness Results

In this section we show hardness for the problems studied in the previous section. All the lower bounds are under projection reductions. We will thus given a circuit C build a graph G with edges labeled by literals, i.e. variables or negations of variables. Given an input x, let $G(x)$ be the graph obtained from G by keeping exactly the edges labeled by literals that are 1 under the assignment x. We then show that $C(x) = 1$ if and only if the graph $G(x)$ satisfies the graph property under consideration.

Our NC^1 lower bounds for the non-planar case build the characterization of NC^1 in terms of permutation branching programs by Barrington [4]. His construction gives for any polynomial size NC^1 circuit a polynomial length *program* over the group S_5 of permutations of 5 elements. From a program of length l we can construct a graph with vertices placed on the grid $\{1, \ldots, l+1\} \times \{1, \ldots, 5\}$. Between two layers of the grid we have 10 edges corresponding to the two permutations corresponding to an instruction of the program. These are then labeled by the corresponding variable or its negation accordingly. The resulting graph $G(x)$ will for any input consist of exactly 5 disjoint paths, and we can without loss of generality assume these are as shown in Figure 5.2 (a) and (b). We can without loss of generality assume that the length $l+1$ is even or odd if needed. We shall call this graph the Barrington graph.

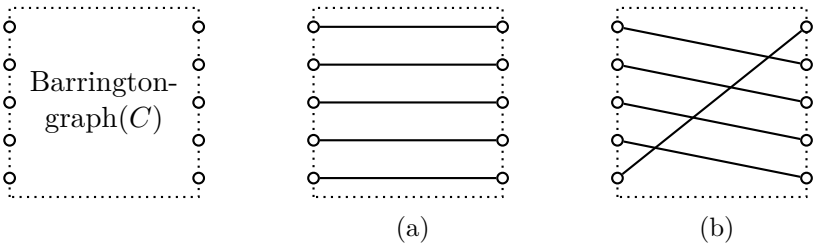
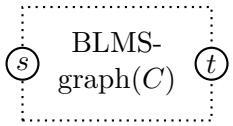
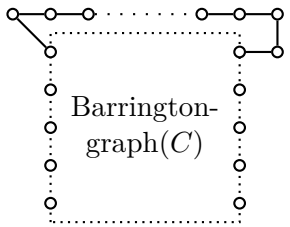


Figure 5.2: Graph for a NC^1 circuit C and paths when $C(x) = 0$ (a) and $C(x) = 1$ (b).

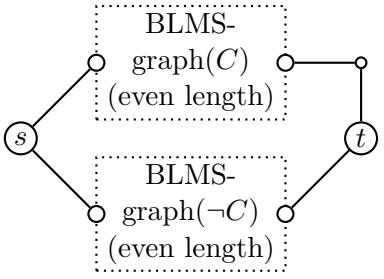


Barrington et al. [5] showed that connectivity in constant width grid graphs is complete for AC^0 under projection reductions. This holds for both undirected graphs and directed graphs. Thus for any AC^0 circuit C we have a grid graph G with edges labeled by literals, with a vertex s in the first layer, a vertex t in the last layer, such that $G(x)$ has a (s, t) path if and only if $C(x) = 1$. We can also here without loss of generality assume the length of the grid graph is even or odd if needed. We will call this graph the BLMS graph.

2-coloring



The results here are based upon the simple fact that a cycle can be 2-colored if and only if it is of even length, and that if a path is 2-colored then the endpoints must have different color if and only if the path is of odd length. In the non-planar case we obtain NC^1-hardness simply by taking the Barrington graph and connecting the top nodes on the two sides by a path whose length is of opposite parity of the length of the graph. Then if these nodes are connected in the Barrington graph an odd cycle appears, and this happens exactly when $C(x) = 0$. Otherwise the graph consists of disjoint paths and can thus be 2-colored.



We next consider the planar case. From the BLMS-graph for a given AC^0 circuit C we construct a 2-coloring gadget graph. The graph is always 2-colorable (since the BLMS graph is bipartite) and has even length. Also if $C(x) = 1$ then s and t must have different colors in any 2-coloring, since in that case there is a path of odd length (through the BLMS graph for C) between s and t. Similarly, if $C(x) = 0$, then s and t must have the same color in any 2-coloring, since in that case there is a path of even length (through the BLMS graph for $\neg C$) between s and t. Consider next a $\mathsf{XOR} \circ \mathsf{AC}^0$ circuit C. Suppose that $C = \mathrm{XOR}(C_1, \ldots, C_m)$ where C_1,...,C_m are AC^0 circuits. We simply concate-

nate the 2-coloring gadgets for $C_1, \ldots, C_m$ as shown in figure 5.3 and connect the s terminal of the first gadget graph with the t terminal of the last gadget graph by an odd length path.

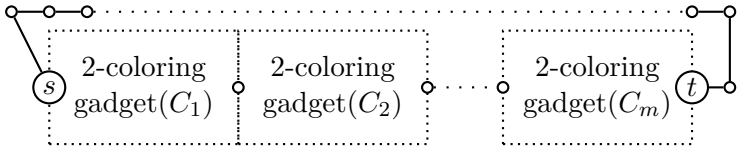
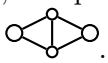
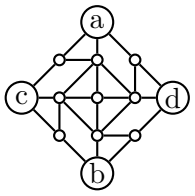


Figure 5.3: Graph for $\mathrm{XOR}(C_1, \ldots, C_m)$, where $C_1, \ldots, C_m$ are AC^0 circuits.

By the property of the gadget graphs, in any 2-coloring, the vertices s and t must have different color exactly when an odd number of the circuits $C_1, \ldots, C_m$ evaluate to 1, and must have the same color otherwise. Since the top path connecting s and t has odd length it follows that the graph can be 2-colored exactly when $\mathrm{XOR}(C_1(x), \ldots, C_m(x)) = 1$. Now given an $\mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AC}^0$ circuit $C = \mathrm{AND}(C_1, \ldots, C_m)$ we construct the above graph for each of the $\mathsf{XOR} \circ \mathsf{AC}^0$ sub-circuits consider the graph that is the disjoint union of all these graphs. Then this graph can be 2-colored if and only if $C(x) = 1$.

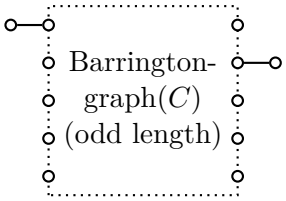
3-coloring

To show NC^1-hardness in the non-planar case we can adapt the graph constructed for the case of 2-coloring, and change it appropriately. Namely we just replace each edge, except for the rightmost vertical edge, by the simple 4-vertex gadget graph  . The gadget graph ensures that all of the original vertices must be of the same color, and hence a coloring is not possible because of the rightmost vertical edge if a cycle is formed in the original graph.

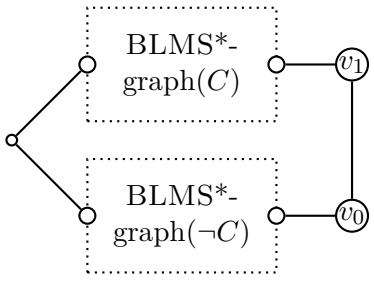


We can transform this graph into a grid graph with diagonals by using the simple crossover gadget as shown on the right that was constructed by Garey et al. [26] for showing that the 3-coloring problem for general planar graphs is NP-complete. This gadget simulates by a grid graph with diagonals a crossing between edges (a, b) and (c, c). Since the graph we start with is layered we can deal with the intersections in each layer separately. By appropriately placing a number of crossover gadgets we can simulate the crossings between the layers by connecting vertices of the surrounding layers to the crossing gadgets. To make the entire graph be a grid graph with diagonals we can finally replace such edges by concatenations of the simple 4-vertex gadget graph together with a regular edge. Doing this shows that 3-coloring remains NC^1-hard for constant width grid graphs with diagonals.

Matching



The results here are based upon the simple fact that a path has a perfect matching if and only if it is of odd length. In the non-planar case we obtain NC^1-hardness simply by taking the Barrington graph of odd length and attaching additional edges to the top vertex on the left side and the second vertex from the top on the right hand side. Then the graph has a perfect matching if and only if these two vertices are connected by a path through the Barrington graph, which in turn happens if and only if $C(x) = 1$. We next consider the planar case. As in the case of 2-coloring we shall for an AC^0 circuit C build a gadget based on the BLMS graph. But we shall first make a modification to the BLMS graph to make it suited for the purpose of a gadget for matching.



We start with the directed version of the BLMS graph. We shall also assume it to be of even length. From this we shall construct an undirected graph we denote by BLMS*. It is obtained by performing the standard reduction from directed (s, t) connectivity to perfect matching. Namely, we split each vertex u into two vertices u and u', connected by an edge. Then for every directed edge from u to v we connect u' and v. We remove the vertices s and t'. The graph always has a partial matching where only s' and t are left unmatched. Furthermore, the graph has a perfect matching if and only if $C(x) = 1$. We can do this preserving planarity and such that the resulting graph has even length. Note also that is a bipartite graph, and we therefore obtain AC^0-hardness for perfect matching on bipartite grid-layered planar graphs. With some more work one can also obtain an equivalent grid graph.

From the BLMS* graphs we construct a matching gadget graph for the circuit C. Let us call all vertices except for v_1 and v_0 for internal vertices. In this graph there is always a partial matching that matches all internal vertices and exactly one of v_0 or v_1. Also if $C(x) = 1$ then any partial matching that matches all internal vertices must leave v_1 unmatched. Similarly, if $C(x) = 0$ then any partial matching that matches all internal vertices must leave v_0 unmatched. We construct the matching gadget graph in such a way it is of even length. Consider next a $\mathsf{XOR} \circ \mathsf{AC}^0$ circuit C. Suppose that $C = \mathrm{XOR}(C_1, \ldots, C_m)$ where $C_1, \ldots, C_m$ are AC^0 circuits and assume without loss of generality that m is even. We place the matching gadgets for $C_1, \ldots, C_m$ adjacent to each other as shown in Figure 5.4. We also have a top path ending in a terminal v_1 and a bottom path ending in a terminal v_0 along the gadgets. The top path is of constructed to be of odd length and the bottom path of even length. The v_1 terminal of a gadget graph is connected to the

top path and the v_0 is connected to the bottom path. They may thus "steal" a vertex from either the top or bottom path depending on the unmatched terminal. In other words, in order to match all the vertices of a gadget for C_i, in case $C_i(x) = 1$ the gadget must steal a vertex from the top path, and in case $C_i(x) = 0$ the gadget must steal a vertex from the bottom path. We can see that the combined graph always have a partial matching where all vertices except exactly one terminal is matched. If $C(x) = 1$ an odd number of vertices are stolen from both the top and bottom path. Thus if all vertices besides the terminals are matched then v_1 is unmatched and v_0 is matched. Similarly, if $C(x) = 0$ an even number of vertices are stolen from both the top and bottom paths. Thus if all vertices besides the terminals are matched then v_1 is matched and v_0 is unmatched. Thus for C we get a matching gadget similar to the matching gadget constructed for AC^0 circuits.

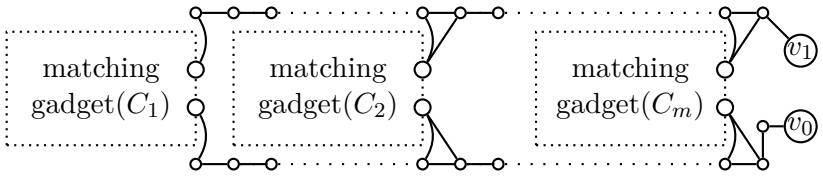


Figure 5.4: Graph for XOR($C_1, \ldots, C_m$), where C_1,...,C_m are AC^0 circuits (even m).

As opposed to the case of 2-coloring we can here go a step further. Consider now an $\mathsf{OR} \circ \mathsf{XOR} \circ \mathsf{AC}^0$ circuit C. Suppose that $C = \text{XOR}(C_1, \ldots, C_m)$ where C_1,...,C_m are $\mathsf{XOR} \circ \mathsf{AC}^0$ circuits and assume without loss of generality that m is even. We place the matching gadgets for $C_1, \ldots, C_m$ adjacent to each other as shown in Figure 5.5. As before we have a top path and a bottom path along the gadgets. Both of the paths are of even length. Different to before, the gadgets are constructed in such a way that if the terminal v_1 is unmatched the gadget may steal a vertex from *either* the top path or the bottom path. If the terminal v_0 is unmatched the gadget may just steal a vertex from the bottom path. Since both the top and bottom path are of even length, the only way to match all vertices is that the gadgets steal an odd number of vertices from both paths. Now if $C(x) = 1$ at least one gadget can be matched such that v_1 is left unmatched. Thus we may pick an odd number of subcircuits C_i to steal a vertex from the top path, and let the remaining steal a vertex from the bottom path. On the other hand if all $C_i(x) = 0$ then to match all vertices of each gadget they need to steal a vertex from the bottom path, meaning that the full graph does not have a perfect matching.

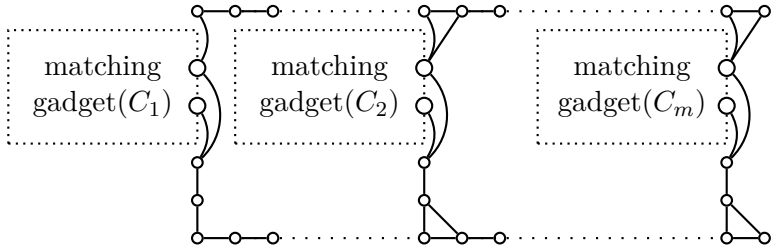
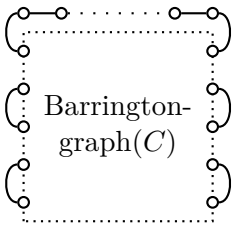


Figure 5.5: Graph for $\mathrm{OR}(C_1, \ldots, C_m)$, where C_1,...,C_m are $\mathsf{XOR} \circ \mathsf{AC}^0$ circuits (even m).

As in the case of 2-coloring, if we construct the final graph to be a disjoint union of such graphs, the matching problem on the resulting graph simulates $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{XOR} \circ \mathsf{AC}^0$ circuits.

Hamiltonian cycle



In the non-planar case we obtain NC^1-hardness simply by taking the Barrington graph, add another path on top, and connecting the nodes on the two sides in pairs. If $C(x) = 0$ the resulting graph consists of 3 disjoint cycles, whereas if $C(x) = 1$ the resulting graph consists of a single cycle. We can not similarly to the case of 3-coloring translate this directly to the planar case, since we have no general crossover gadget. We can however proceed following the approach shown for establishing NP-completeness for the case of general planar graphs. We shall just outline the proof. Plesńik [42] showed that the Hamiltonian Cycle problem over directed planar graphs of degree 2 is NP-complete. Our main observation is that we can use the same reduction to reduce the complement of directed connectivity the Hamiltonian Cycle problem on bounded width planar graphs. Given a graph G with source s and target t we derive a 2-CNF formula F consisting of the following clauses: (s), $(\neg t)$ and for each directed edge $(u \to v)$ there is a clause $(\neg u \vee v)$. It follows that F is satisfiable if and only if there is no path in G from s to t. We then apply the reduction in [42] to F. We observe that the resulting graph is planar and bounded width. To see this, we note that clauses in F can be ordered so that for every node u, the set of clauses containing u appear in an interval of constant length. This order essentially follows the order according to G. After applying the reduction to F at some point a graph with crossings is obtained. But because of the interval property of F all crossings can appear in such intervals of constant length, which means that we can apply the crossing gadget to each such interval and blow up the width by only a constant. To get a grid graph we then apply a reduction from [35]. We first transform the graph obtained in the first part of the reduction to a bipartite graph and then embed it into a grid. However we should note that [35] embeds a n-vertex graph into a $\Theta(n) \times \Theta(n)$

grid. Since the graph that we start with is layered, we can apply the embedding on consecutive layers separately and hence we will get a constant width grid graph.

Disjoint paths

Here we need just remark that the disjoint paths problem is precisely the connectivity problem when $k = 1$. Thus we have NC^1-hardness in the non-planar case by the result of Barrington [4] and AC^0-hardness in the planar case by the result of Barrington et al. [5].

Chapter 6

Open Problems

This thesis leaves several questions unanswered, both in the area of Boolean satisfiability and the complexity of planar problems. We list a few immediate ones which naturally arise from our work.

- **Sharp lower bounds for PPSZ and DPLL**: We gave constructions of hard instances of k-CNF formulas for PPSZ and DPLL which require running times of the form $2^{(1-\epsilon_k)n}$, where $\epsilon_k = O(\log^2 k/k)$ for the former, and $\epsilon_k = \widetilde{O}(k^{-1/3})$ for the latter. On the other hand for both algorithms we have $\epsilon_k = \Omega(1/k)$, and hence there remains a big gap. It seems to be a challenging problem to close this gap, however it seems to be a worthwhile direction, especially for DPLL, as the gap is larger and that improvements can potentially give rise to new techniques in proof complexity.

- **Depth vs variable space in resolution**: We showed that if a formula has small variable space refutations, then it could also be refuted in small depth. Our proof cannot deal with the case when variables space is $\omega(\log \log n)$. Is there a large gap between depth and variable space in this regime of parameters?

- **The number of prime implicant of k-CNFs**: Is it true that for large enough k, any k-CNF formula has at most $3^{(1-\Omega(1/k))n}$ prime implicants? We could only verify this for read-constant formulas.

- **Sharp bounds for bounded-width graph problems**: We gave an ACC^0 upper bound for the perfect matching problem over bipartite grid-layered graphs. Is it possible to improve this bound to AC^0? This would require a further analysis of groups inside the monoid defined for the perfect matching problem. Another problem is to tighten the bound for 2-coloring over grid-planar graphs. Is there a projection reduction from $\mathsf{AC}[2]$ to this problem?

Bibliography

[1] Eric Allender and Ulrich Hertrampf. Depth reduction for circuits of unbounded fan-in. *Information and Computation*, 112(2):217–238, 1994. 45

[2] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. 35

[3] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008. 31

[4] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci*, 38(1):150–164, 1989. 4, 45, 60, 66

[5] David A. Mix Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. Searching constant width mazes captures the AC^0 hierarchy. In *STACS*, volume 1373 of *Lecture Notes in Computer Science*, pages 73–83. Springer, 1998. 4, 61, 66

[6] David A. Mix Barrington and Howard Straubing. Superlinear lower bounds for bounded-width branching programs. *J. Comput. Syst. Sci.*, 50(3):374–381, 1995. 4

[7] David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, 1988. 4, 47, 48

[8] Paul Beame. A Switching Lemma Primer. Technical report, Department of Computer Science and Engineering, University of Washington, 1994. UW-CSE-95-07-01. 9

[9] Paul Beame, Richard M. Karp, Toniann Pitassi, and Michael E. Saks. On the complexity of unsatisfiability proofs for random k-cnf formulas. In *STOC*, pages 561–571, 1998. 31

[10] Christopher Beck and Russell Impagliazzo. Strong ETH Holds for Regular Resolution. In *Proceedings of the Forty-fifth Annual ACM Symposium on*

Theory of Computing, STOC '13, pages 487–494, New York, NY, USA, 2013. ACM. 3, 8, 9, 10, 12

[11] Eli Ben-Sasson and Russell Impagliazzo. Random cnf's are hard for the polynomial calculus. *Computational Complexity*, 19(4):501–519, 2010. 3

[12] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001. 3, 13, 31

[13] Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A characterization of tree-like Resolution size. *Inf. Process. Lett.*, 113(18):666–671, 2013. 10

[14] H. L. Bodlaender. Some classes of planar graphs with bounded treewidth. *Bulletin of the EATCS*, 36:116–126, 1988. 44

[15] Ilario Bonacina and Navid Talebanfard. Improved strong ETH lower bounds for tree-like resolution. submitted. 7

[16] Ashok K. Chandra and George Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24(1):7– 11, 1978. 35, 39

[17] Shiteng Chen, Dominik Scheder, Navid Talebanfard, and Bangsheng Tang. Exponential lower bounds for the PPSZ k-SAT algorithm. In *SODA*, 2013. 17

[18] Vasek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988. 31

[19] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *FOCS*, pages 197–206. IEEE Computer Society, 2013. 44

[20] Samir Datta, Arjun Gopalan, Raghav Kulkarni, and Raghunath Tewari. Improved bounds for bipartite matching on surfaces. In *STACS*, volume 14 of *LIPIcs*, pages 254–265. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. 44

[21] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. i, iii, 2

[22] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960. i, iii, 2, 7

[23] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Algorithmic meta theorems for circuit classes of constant and logarithmic depth. In *STACS*, volume 14 of *LIPIcs*, pages 66–77. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. 44, 45

[24] Walter Feit and John G. Thompson. Solvability of groups of odd order. *Pacific J. Math.*, 13(3):775–1029, 1963. 53

[25] Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci*, 10:111–121, 1980. 44

[26] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci*, 1(3):237–267, 1976. 44, 62

[27] M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM J. Comput*, 5(4):704–714, 1976. 44

[28] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985. 3, 31

[29] Kristoffer Arnsfelt Hansen and Michal Koucký. A new characterization of ACC^0 and probabilistic CC^0. *Computational Complexity*, 19(2):211–234, 2010. 4

[30] KristofferArnsfelt Hansen, Balagopal Komarath, Jayalal Sarma, Sven Skyum, and Navid Talebanfard. Circuit complexity of properties of graphs with constant planar cutwidth. In *Mathematical Foundations of Computer Science 2014*, volume 8635 of *Lecture Notes in Computer Science*, pages 336–347. Springer Berlin Heidelberg, 2014. 4, 43

[31] Timon Hertli. 3-sat faster and simpler - unique-sat bounds for ppsz hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014. 17

[32] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *SODA*, pages 961–972, 2012. 2

[33] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. 2

[34] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. 2

[35] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput*, 11(4):676–686, 1982. 44, 65

[36] P. W. Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967. 44

[37] Peter Bro Miltersen, Jaikumar Radhakrishnan, and Ingo Wegener. On converting CNF to DNF. *Theor. Comput. Sci.*, 347(1-2):325–335, 2005. 8, 10

[38] J.W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965. 38

[39] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM J. Comput.*, 39(1):59–121, 2009. 31

[40] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005. i, iii, 2, 16

[41] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *FOCS*, pages 566–574, 1997. i, 1, 2, 14, 15, 16, 35, 37, 39

[42] Ján Plesník. The NP-completeness of the hamiltonian cycle problem in planar digraphs with degree bound two. *Inf. Process. Lett*, 8(4):199–201, 1979. 65

[43] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *SODA*, pages 128–136, 2000. 3, 8

[44] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *SODA*, pages 128–136, 2000. 3

[45] Alexander A. Razborov. Lower bounds for the size of circuits of bounded depth with basis ($\wedge$, $\oplus$). *Mathematical Notes of the Academy of Science of the USSR*, 41(4):333–338, 1987. 45

[46] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008. 44

[47] Neil Robertson and Paul D. Seymour. Graph minors.XIII. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. 44

[48] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *FOCS*, pages 183–192, 2010. 2

[49] Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM J. Comput*, 23(4):780–788, 1994. 44

[50] Robert H. Sloan, Balázs Szörényi, and György Turán. On k-term DNF with the largest number of prime implicants. *SIAM J. Discrete Math.*, 21(4):987–998, 2008. 35

[51] Navid Talebanfard. On the structure and the number of prime implicants of 2-CNFs. submitted. 36

[52] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. 3, 31

[53] Alasdair Urquhart. The depth of resolution proofs. *Studia Logica*, 99(1-3):349–364, 2011. i, iii, 3, 31, 32

[54] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, pages 162–176, 1977. 1

[55] Vijay V. Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Inf. Comput*, 80(2):152–164, 1989. 44

[56] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009. 1

[57] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *STOC*, pages 231–240, 2010. 1

[58] Ryan Williams. Non-uniform ACC circuit lower bounds. In *IEEE Conference on Computational Complexity*, pages 115–125, 2011. 1