TEL AVIV UNIVERSITY
THE RAYMOND AND BEVERLY SACKLER FACULTY OF EXACT SCIENCES
SCHOOL OF COMPUTER SCIENCE

STUDIES IN ALGEBRAIC AND PROPOSITIONAL PROOF COMPLEXITY

Thesis Submitted for the Degree of Doctor of Philosophy

by

Iddo Tzameret

Under the supervision of
Professor Nachum Dershowitz and Professor Ran Raz

SUBMITTED TO THE SENATE OF TEL AVIV UNIVERSITY

AUGUST 2008

This work was carried out under the supervision of
Prof. Nachum Dershowitz (Tel Aviv University) and **Prof. Ran Raz** (Weizmann Institute)

Contents

List of Figures

Abstract

The field of proof complexity aims at characterizing which statements have short proofs in a given formal proof system. This thesis is a contribution to proof complexity broadly construed as the field that studies the sizes of *structured* or *symbolic* proofs. Our focus will be on the development and complexity-theoretic study of new frameworks, mainly of an algebraic nature, for providing, among other things, proofs of propositional tautologies. We further link and motivate the proof systems we explore with certain questions, mainly from algebraic complexity. The main results of this thesis can be divided into four parts, as follows.

MULTILINEAR PROOFS: We introduce an algebraic proof system that operates with multilinear arithmetic formulas. We show that this proof system is fairly strong, even when restricted to multilinear arithmetic formulas of a very small depth. Specifically, we show that algebraic proofs manipulating depth-2 multilinear arithmetic formulas polynomially simulate resolution, Polynomial Calculus (PC) and Polynomial Calculus with Resolution (PCR) proofs. We provide polynomial size proofs manipulating depth-3 multilinear arithmetic formulas for the pigeonhole principle tautologies and the Tseitin's graph tautologies.

By known lower bounds, this demonstrates that algebraic proof systems manipulating depth-3 multilinear formulas are strictly stronger than resolution, PC and PCR, and have an exponential gap over bounded-depth Frege for both the pigeonhole principle and Tseitin's graph tautologies.

We illustrate a connection between lower bounds on multilinear proofs and lower bounds on multilinear circuits. In particular, we show that (an explicit) super-polynomial size separation between proofs manipulating *general* arithmetic circuits and proofs manipulating *multilinear* circuits implies a super-polynomial size lower bound on multilinear circuits for an explicit family of polynomials.

The short multilinear proofs for hard tautologies are established via a connection between depth-3 multilinear proofs and extensions of resolution, described as follows:

RESOLUTION OVER LINEAR EQUATIONS WITH APPLICATIONS TO MULTILINEAR PROOFS: We develop and study the complexity of propositional proof systems of varying strength extending resolution by allowing it to operate with disjunctions of linear equations instead of clauses. We demonstrate polynomial-size refutations for hard tautologies like the pigeonhole principle, Tseitin graph tautologies and the clique-coloring tautologies in these proof systems. Using (monotone) interpolation we establish an exponential-size lower bound on refutations in a certain, strong, fragment of resolution over linear equations, as well as a general polynomial upper bound on (non-monotone) interpolants in this fragment. We show that proofs operating with depth-3 multilinear formulas polynomially simulate a certain, strong, fragment of resolution over linear equations (by which the aforementioned upper bounds on multilinear proofs follow). We then connect resolution over linear equations with extensions of the cutting planes proof system.

SYMBOLIC PROOFS OF POLYNOMIAL IDENTITIES: To transform algebraic propositional proof systems operating with arithmetic formulas into formal proof systems one usually augments the system with an "auxiliary" proof system capable of manipulating arithmetic formulas by means of the polynomial-ring axioms. We investigate basic structural and complexity characterizations of the latter proof system and its fragments. Specifically, a *symbolic proof* for establishing that a given arithmetic formula Φ computes the zero polynomial (or equivalently, that two given arithmetic formulas compute the same polynomial) is a sequence of formulas, starting with Φ and deriving the formula 0 by means of the standard polynomial-ring axioms applied to any subformula. We introduce fragments of symbolic proofs named *analytic symbolic proofs*, enjoying a natural property: a symbolic proof is analytic if one cannot introduce arbitrary new formulas anywhere in the proof (that is, formulas computing the zero polynomial which do not originate, in a precise manner, from the initial arithmetic formula). We establish exponential lower bounds on the number of steps in analytic symbolic proofs operating with depth-3 arithmetic formulas, under a certain regularity condition on the structure of proofs (roughly, mimicking a tree-like structure). The hard instances are explicit and rely on small formulas for the symmetric polynomials.

ALTERNATIVE MODELS OF REFUTATION – PROMISE PROPOSITIONAL PROOFS: We study the problem of certifying unsatisfiability of CNF formulas under the promise that any satisfiable formula has many satisfying assignments, where "many" stands for an explicitly specified function Λ in the number of variables n. To this end, we develop propositional proof systems under different measures of promises (that is, different Λ) as extensions of resolution. This is done by augmenting resolution with axioms that, roughly, can eliminate sets of truth assignments defined by Boolean circuits. We then investigate the complexity of such systems, obtaining an exponential separation in the average-case between resolution under different size promises: (i) Resolution has polynomial-size refutations for all unsatisfiable 3CNF formulas when the promise is $\varepsilon \cdot 2^n$, for any constant $0 < \varepsilon < 1$; (ii) There are no sub-exponential size resolution refutations for random 3CNF formulas, when the promise is $2^{\delta n}$ (and the number of clauses is $O(n^{3/2-\epsilon})$, for $0 < \epsilon < \frac{1}{2}$), for any constant $0 < \delta < 1$.

Acknowledgements

First and foremost I would like to thank my supervisors Nachum Dershowitz and Ran Raz. I went with Nachum a long way since the days before I started my graduate studies, and I am greatly indebted to his continuous support, interest, generosity, kindness and also his great advices and knowledge through all of this time. Ever since I met Ran, when I began my graduate studies, his deep understanding and brilliant insights gave a tremendous push to my research and were invaluable in the course of my work. I was very fortunate to have him as my advisor and I'm indebted to his great patience which led us to rather lengthy talks and for his commitment to research which had a great impact on me. Ran constantly encouraged me to develop my research and to find new directions. His contribution to this thesis is invaluable in both his ideas and his strive toward complete thoroughness and clarity.

A very special thanks goes to Jan Krajíček for inviting me to the Cambridge workshop on proof complexity, for the excellent yearly fall schools organized by him, and for his interest in my work as well as his very useful comments. I believe that Jan's book on bounded arithmetic and proof complexity was one of the main reasons I got interested in proof complexity in the first place.

I would also like to thank my friends and colleagues from the open-space, and especially Amir Epstein and Zur Izhakian; and also colleagues with whom I met in numerous proof-complexity gatherings, and in particular Pavel Pudlák and other members and guests of the "Prague school", among these are Marta Bílková, Emil Jeřábek, Leszek Kołodziejczyk, Phuong Nguyen, Jakob Nordström, Steven Perron and Neil Thapen.

I had very helpful discussions with Pavel Hrubeš and Neil Thapen on some of the results in this thesis. For some results I've also benefited from discussions with Amnon Ta-Shma, whose versatile complexity courses were also of great help; and helpful correspondence with Eli Ben-Sasson, Dima Grigoriev and Arist Kojevnikov. Thanks also to Sam Buss for sending me a Xerox of his (out of print) seminal book. I thank Michael Ben-Or and Uri Zwick for serving in my thesis proposal committee, and Arnon Avron, Udi Boker, Yoram Hirshfeld, Alex Rabinovich, Boaz (Boris) Trakhtenbrot and Anna Zamansky for the opinionated logic seminars. Anonymous referees of some of my papers were of great help in improving the exposition of some of the material in this thesis. During my studies I was supported by the Israel Science Foundation (grant no. 250/05). Last but not least, I wish to thank my parents for their constant support during the years of research.

Iddo Tzameret August 2008

Tel Aviv, Israel

Chapter 1

Introduction

Proof complexity lies at the intersection of computational complexity and logic. Computational complexity aims at understanding the nature and limitations of efficient computation; logic, on the other hand, and in particular proof theory, studies what can formally be proved from a given set of axioms and deduction rules, in a given language. Accordingly, *proof complexity* aims at characterizing which statements can formally be proved with *efficient* (or feasible) proofs. Here the term "efficient" typically stands for two separate (though interconnected) meanings: the first meaning refers to the size of the proofs, that is, the number of symbols (or bits) it takes to write down the proofs. The second meaning refers to the efficiency of the "concepts" one reasons with (the technical interpretation being the complexity class from which proof-lines are taken).

Let us begin with a short introduction to the field of *propositional* proof complexity.

1.1 Background on Propositional Proof Complexity

Propositional proof complexity deals with proof systems that establish *propositional* tautologies. A propositional proof system is usually described by a finite set of inference rules and axiom schemata. A propositional proof is then a derivation of some tautology that applies the prescribed inference rules to the set of axioms. We can sometimes take the dual view in which proof systems establish that some formula is unsatisfiable by deriving FALSE from the formula and axioms. Thus, the proofs in such systems are usually called *refutations*.

The standard definition of a *formal* (or *abstract*) proof system in this setting is that introduced by Cook and Reckhow in their seminal work Cook and Reckhow (1979).

Definition 1.1.1 (Cook-Reckhow propositional proof system) *A* Cook-Reckhow propositional proof system *(or a* formal *propositional proof system) is a polynomial-time algorithm A that receives a propositional formula F (most commonly a Boolean formula in the connectives $\vee, \wedge, \neg$) and a string π over some finite alphabet ("the [proposed] proof" of F), such that there exists a π with $A(F, \pi) = 1$ if and only if F is a tautology.*

Note that the requirement of a formal proof system A is that proofs are polynomial-time recognizable (polynomial in the size of both the *proofs* π and the formulas F). The *completeness* of a (Cook-Reckhow) proof system (with respect to the set of all propositional formulas; or for a subset of it, e.g. the set of tautological formulas in disjunctive normal form [DNFs]) means that *every* tautological formula F has a string π ("the proof of F") so that $A(F, \pi) = 1$. The *soundness* of a (Cook-Reckhow) proof system means that for every formula F, if there is a string π for which $A(F, \pi) = 1$, then F is indeed a tautology.

Given Definition 1.1.1, the basic question of propositional proof complexity is this:

> *Fix a formal propositional proof system $\mathcal{P}$ and a propositional tautology τ. What is the smallest proof of τ in $\mathcal{P}$?*

From the perspective of *computational complexity*, this question is of fundamental importance, since showing that for every propositional proof system $\mathcal{P}$, there are tautologies τ with no polynomial-size (in the size of τ) proofs in $\mathcal{P}$, would readily imply $\mathbf{NP} \neq \mathbf{coNP}$, as was observed in Cook and Reckhow (1979).

From the perspective of *algorithmics* (and computational logic), establishing lower bounds on the size of proofs in certain proof systems, usually gives lower bounds on the run-time of specific procedures for **NP**-complete problems like certain SAT-solvers and theorem provers (cf. Impagliazzo et al. (1999); Beame et al. (2002); Segerlind (2007a)). These run-time lower bounds are rather broad, in the sense that they do not depend on specific heuristics taken by the procedures for solving the problems.

Furthermore, from the perspective of *mathematical logic*, and specifically the proof- and model-theoretic study of formal (first, or higher-order) systems of arithmetic, tight and significant connections with propositional proof complexity were discovered. To some extent, propositional proofs are the non-uniform counterpart of weak formal systems of arithmetic (generally called theories of *bounded arithmetic*); in fact, lower bounds on propositional proof systems are closely related to independence (that is, unprovability) results in weak systems of arithmetic (cf. Buss (1986); Krajíček (1995); Razborov (1996); Buss (1997); Cook (2005); Cook and Nguyen (2004–2008)).

Yet another aspect of propositional proof complexity is its tight relation – in spirit and techniques – to *circuit complexity*. It seems that there is a correspondence between proving lower bounds on a circuit class and proving lower bounds on proofs operating with circuits of the prescribed circuit class, though formal and specific relations between proof complexity hardness and computational hardness are quite restricted currently (see Beame and Pitassi (1998) for a short discussion on circuit-complexity based proof systems; see also Segerlind (2007b) for a technical survey of propositional proof complexity).

In what follows we focus on concrete proof systems that are usually studied in the proof complexity literature and which are relevant to this thesis.

1.1.1 Concrete Proof Systems

Perhaps the most natural and typical family of proof systems are those systems originating in mathematical logic, which are called *Frege proof systems* (or Hilbert-style proof systems). When considering only the *propositional* fragment of this family of proof systems, we obtain a textbook proof system, in which each proof-line[1] is usually a Boolean formula with the logical connectives $\wedge, \vee, \neg$, which stand for AND, OR, NOT, respectively (and, optionally, the implication connective $\rightarrow$). A proof in a Frege system is a sequence of formulas that starts from a set of self-evident (and easily recognizable) axioms and derives new formulas (that is, new proof-lines) by using a set of self-evident deduction rules. In this way, the proof system enables one to derive any propositional tautology (and only propositional tautologies).

[1]Each element (usually a formula) of a proof-sequence is referred to as a *proof-line*.

Algebraic proof systems. In the course of investigating the complexity of different propositional proof systems, connections were found between proofs operating with Boolean formulas and proofs operating with multivariate polynomials over a fixed field (cf. Beame et al. (1996); Pitassi (1997)). Proof systems operating with polynomials are called *algebraic proof systems*. In this setting one replaces logical reasoning (as used in Frege proofs), with basic algebraic reasoning. Specifically, in algebraic proofs one usually starts from a set of polynomials, with the intended semantics that each of the polynomials evaluates to 0 over the field, and by using simple algebraic deduction rules like addition of two polynomials and multiplication of a polynomial by a variable, derives new polynomials (from previous ones). Such algebraic proofs usually demonstrate that a collection of polynomial equations has no solutions (that is, no common roots) over some fixed field. When the collections of initial polynomial equations are derived from propositional contradiction (most typically, from the clauses of unsatisfiable formulas in conjunctive normal form [CNF]) and the algebraic proofs establish that the collections have no $0, 1$ solutions over the field, then the proof system is called an *algebraic propositional proof system*.

The Polynomial Calculus proof system (PC, for short), introduced in Clegg et al. (1996), is a well studied algebraic (propositional) proof system. Fix some field $\mathbb{F}$ and let Q be a collection of multivariate polynomial equations $Q_1 = 0, \ldots, Q_m = 0$, where each Q_i is taken from the ring of polynomials $\mathbb{F}[x_1, \ldots, x_n]$. In PC the fact that the collection Q has no $0, 1$ solutions over the field $\mathbb{F}$ is proved by using the following basic algebraic inference rules: from two polynomials p and q (interpreted as the two equations $p = 0$ and $q = 0$) we can deduce $\alpha \cdot p + \beta \cdot q$, where α, β are elements of $\mathbb{F}$; and from p we can deduce $x_i \cdot p$, for any variable x_i $(1 \leq i \leq n)$. A sequence of polynomials that uses $Q_1, \ldots, Q_m$ and $x_i^2 - x_i$ (for any variable x_i) as initial polynomial equations, follows the above algebraic inference rules, and terminates with 1 (with the intended semantics being that 1 evaluates to 0; thus, 1 here stands for FALSE), is called a PC *refutation* of the polynomial (equations) $Q_1, \ldots, Q_m$.

It can be shown by Hilbert's Nullstellensatz that for every unsatisfiable set of polynomials (*unsatisfiable*, in the sense that the polynomials have no common $0, 1$ roots over the field) there exists a PC refutation. Also, since the deduction rules are sound, every derivation of the polynomial 1 must start from a set of unsatisfiable set of polynomials, and so overall we obtain a sound and complete proof system that can establish the unsatisfiability of every unsatisfiable set of polynomials over the field.

When we consider algebraic proof systems we have the possibility to relax somewhat the notion of a *proof*: instead of requiring that a proof be polynomially-verifiable (as is required by the definition of a formal proof system), we may only require that the proof will be polynomially-verifiable with *high probability*. This is done by considering the proofs as being *semantic proofs* (this semantic setting was first considered by Pitassi in Pitassi (1997)). In other words, each proof-line can be regarded as (the set of solutions of) a (formal) multivariate polynomial over the given field; and where each such polynomial is written as an arbitrarily chosen arithmetic formula computing the polynomial (note that a single polynomial might have many representation as an arithmetic formula). Some of the algebraic proof systems we study in this thesis are semantic proof systems. We shall also study ways to turn semantic algebraic proofs into syntactic ones (that is, into formal propositional proof systems – verifiable in deterministic polynomial time).

Besides the fact that algebraic proof systems are almost as natural as their Boolean counterparts (that is, Frege proof systems), there are many other motivations for studying algebraic proofs. In fact, the initial interest in algebraic proof systems arose due to their connections with Frege proof systems operating with constant-depth formulas equipped with modular counting gates and Frege proofs operating with constant-depth formulas augmented with modular counting axioms (cf. Beame et al. (1996); Buss et al. (1996/97); Beame and Riis (1998); Impagliazzo and Segerlind (2001, 2002)). Proving lower bounds on constant depth Frege proof with counting gates is still an important open problem in proof complexity theory. Another motivation for considering algebraic proof system is more akin to questions of practical importance, for instance, boosting performance in automatic theorem provers and SAT-solvers by using algebraic reasoning (see for example Clegg et al. (1996); Hirsch et al. (2005)).

In this thesis we study algebraic proof systems as objects of interest in their own right, and we shall also consider their connections with questions in algebraic complexity and algebraic circuit complexity.

1.1.2 Overview of Proof Systems

We now list the proof (and refutation) systems we consider in this thesis. Some of the proof systems have already been considered in previous works and other systems are introduced in this thesis (we shall define these formally in the relevant places).

Algebraic propositional proof systems. *Polynomial Calculus*, denoted PC (introduced in Clegg et al. (1996)). A proof system for the set of unsatisfiable CNF formulas written as an unsatisfiable set of polynomial equations over a field. Each polynomial in a PC proof is represented as an explicit sum of monomials.

Polynomial Calculus with Resolution, denoted PCR (introduced in Alekhnovich et al. (2002)). This is an extension of PC where for each variable x_i a new formal variable $\bar{x}_i$ is added. The variable $\bar{x}_i$ equals $1 - x_i$. Each polynomial in a PCR proof is represented as an explicit sum of monomials. PCR can polynomially simulate both PC and resolution.

Formula Multilinear Calculus, denoted fMC. The fMC is a semantic algebraic propositional proof system for the set of unsatisfiable CNF formulas written as unsatisfiable set of multilinear polynomial equations over a field. Each polynomial in an fMC proof is a multilinear polynomial represented as a multilinear arithmetic formula (we consider arithmetic formulas that can use unbounded fan-in $+$ (addition) and $\times$ (product) gates).

Depth-k Formula Multilinear Calculus, denoted depth-k fMC. This is a restriction of fMC to multilinear arithmetic formulas of depth at most k.

The cMC *and* cPCR *proof systems*. The cMC proof system is similar to fMC, except that multilinear polynomials are represented by multilinear arithmetic *circuits* (instead of multilinear arithmetic formulas). In the same manner, cPCR is a proof system similar to PCR, except that polynomials are represented by (general) arithmetic circuits (instead of sums of monomials).

Proof systems operating with Boolean formulas. *Resolution* (introduced by Blake (1937) and further developed in Robinson (1965)). A proof system for establishing the unsatisfiability of CNF formulas. Each resolution proof-line consists of a clause (i.e., a disjunction of variables or their negations). The last line of a resolution refutation is the empty clause, which has no satisfying assignment.

Bounded-depth Frege. This system is usually considered as a proof system for the set of Boolean tautologies. The lines in a bounded depth Frege proof consists of *constant-depth* formulas over the connective NOT and the unbounded fan-in connectives AND, OR. We can consider bounded-depth Frege to be also a proof system for the set of *unsatisfiable* Boolean formulas, by treating a proof sequence (starting from some initial set of unsatisfiable formulas) that ends with FALSE, as a refutation.

Propositional proofs under a promise. These are refutation systems operating with Boolean formulas (or clauses, in case we deal with extensions of resolution) that are complete and sound for the set of formulas promised to be either unsatisfiable or to have sufficiently many satisfying assignments (where the term "sufficiently many" stands for an explicitly given function of the number of variables in the formula).

Boolean-Algebraic proof systems. *Resolution over linear equations (*R(lin)*, for short) and its fragments.* The R(lin) proof system establishes unsatisfiable CNF formulas. Each R(lin) proof-line consists of a disjunction of linear equations with integer coefficients (the coefficients are written in unary notation). The last line of an R(lin) refutation is the empty clause, which has no satisfying assignment. We shall also define a (provably weaker) fragment of R(lin), denoted R^0(lin).

Extensions of the cutting planes proof system. The R(CP*) proof system (introduced in Krajíček (1998)) is a common extension of resolution and CP* (the latter is cutting planes with polynomially bounded coefficients). The system R(CP*) is essentially resolution operating with *disjunctions of linear inequalities* (with polynomially bounded integral coefficients) augmented with the cutting planes inference rules.

Algebraic proof systems for polynomial identities. *Symbolic proofs of polynomial identities.* A *symbolic proof* for establishing that a given arithmetic formula Φ computes the zero polynomial (or equivalently, that two given arithmetic formulas compute the same polynomial) is a sequence of formulas, starting with Φ and deriving the formula 0 by means of the standard polynomial-ring axioms applied to any subformula.

1.1.3 Comparing Proof Systems: Simulations and Separations

For two propositional proof systems P_1, P_2 we say that P_2 *polynomially simulates* P_1 if for every propositional formula F and every P_1-proof π of F, there exists a proof of F in P_2 of size polynomial in the size of π. In case P_2 polynomially simulates P_1 while P_1 does not polynomially simulate P_2 we say that P_2 is *strictly stronger* than P_1 or that *there is a (super-polynomial) separation of* P_2 *and* P_1. In case a proof system P uses a language that is different language from the set of Boolean propositional formulas, we shall fix ahead a simple (and efficient) translation that maps Boolean propositional formulas to the language used by P (and this way we can treat the proof system P as a proof system for the set of Boolean propositional formulas).

In this thesis, when comparing the strength of different propositional proof systems, we shall restrict ourselves to CNF formulas only. That is, we consider propositional proof systems such as resolution and bounded-depth Frege as proof systems for the set of unsatisfiable CNF formulas and we consider algebraic proof systems to be proof systems for the set of polynomial translations of unsatisfiable CNF formulas (see Section 2.4). More formally:

Definition 1.1.2 (Simulations and separations) *Let P_1, P_2 be two proof systems for the set of unsatisfiable CNF formulas. We say that P_2* polynomially simulates *P_1 if given a P_1 refutation π of a CNF F, there exists a refutation of F in P_2 of size polynomial in the size of π. In case P_2 polynomially simulates P_1 while P_1 does not polynomially simulate P_2 we say that P_2 is* strictly stronger *than P_1 and also that* there is a (super-polynomial) separation of P_2 and P_1. *Given an unsatisfiable CNF formula F, we say that P_2* has an exponential gap over P_1 for F, *if there exists a polynomial size P_2 refutation of F, and the smallest P_1 refutation of F is of exponential size. If either P_1 or P_2 are* algebraic *proof systems, then we identify the CNF formula F with its translation to a collection of polynomial equations.*

For the sake of convenience we shall sometimes write simply *simulates* to mean *polynomially simulates*. Since we do not talk about other concepts of simulations, there should be no confusion.

1.2 Contribution of the Thesis

The thesis is divided basically into the following four parts: multilinear proofs, resolution over linear equations, symbolic proofs of polynomial identities and propositional proofs under a promise. The following subsections elaborate on these four parts.

1.2.1 Multilinear Proofs

In Chapter 3 we introduce a natural family of algebraic proof systems generally called *multilinear proof systems*. In particular we consider an algebraic proof system Formula

Multilinear Calculus (fMC) that manipulates multilinear arithmetic formulas. A multilinear proof (that is, an fMC proof) begins with an initial set of multilinear polynomial equations representing the clauses of a CNF formula (where a polynomial is multilinear if in each of its monomials the power of every variable is at most one), and the goal is to prove that the CNF formula is unsatisfiable by showing the equations have no $0, 1$ solutions over a given fixed field.

Specifically, let Q be a set of initial multilinear polynomial equations in the formal variables $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ over some fixed field. A multilinear proof of the insolvability of Q is a sequence of multilinear polynomial equations, where *each polynomial is represented as (an arbitrarily chosen) multilinear arithmetic formula* (a multilinear arithmetic formula is an arithmetic formula in which every gate computes a multilinear polynomial). The sequence uses the initial equations plus the polynomial equations $x_i + \bar{x}_i - 1 = 0$ and $x_i \cdot \bar{x}_i = 0$ (for all variables $x_i, \bar{x}_i$) as axioms, and terminates with the unsatisfiable equation $1 = 0$. Derivations of polynomial equations in the sequence are done by applying the following two basic algebraic inference rules to previous equations in the sequence:

- from $p = 0$ and $q = 0$ one can deduce $\alpha \cdot p + \beta \cdot q = 0$, where α, β are elements of the field;
- from $p = 0$ one can deduce $q \cdot p = 0$, for any polynomial q such that $q \cdot p$ is multilinear.

(The inclusion of the equalities $x_i + \bar{x}_i - 1 = 0$ and $x_i \cdot \bar{x}_i = 0$ forces the variables x_i and $\bar{x}_i$ to take on only the Boolean values 0 and 1, where $\bar{x}_i$ takes the negative value of x_i.) If such a sequence exists then there is no assignment of $0, 1$ values that satisfies all the initial equations. Such a proof of insolvability is then called a *multilinear refutation* of the initial polynomial equations.

We can obtain in this way a proof system for (unsatisfiable) CNF formulas. Given a CNF formula F in the variables $x_1, \ldots, x_n$ we translate F to a system of multilinear polynomial equations in the variables $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$. Each clause C of F translates into a multilinear polynomial equation $q_C = 0$. F is satisfiable if and only if the system of polynomial equations $q_C = 0$, for all clauses C of F, has a common root in the field, where the root also satisfies the axioms $x_i + \bar{x}_i - 1 = 0$ and $x_i \cdot \bar{x}_i = 0$ (for all variables $x_i, \bar{x}_i$). For example, the CNF $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_4)$ translates into the polynomial equations $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 = 0$, $x_2 \cdot \bar{x}_4 = 0$.

The minimal refutation size of a given set of initial polynomial equations (i.e., the number of symbols that it takes to write down the refutation of these equations) is the standard measure for the strength of an algebraic proof system. In algebraic proof systems such as the Polynomial Calculus (PC) (described above) and Polynomial Calculus with Resolution (PCR), one represents the polynomials inside refutations as explicit sum of monomials. Then, the size of a PC or a PCR refutation is defined as the total number of all monomials appearing in the refutation. On the other hand, in the multilinear proof system we present, polynomials inside refutations are represented as *multilinear arithmetic* *formulas*. Accordingly, the size of a multilinear refutation is defined to be the total size of all the multilinear arithmetic formulas appearing in the refutation.

Our results show that algebraic proof systems manipulating multilinear arithmetic formulas – and further, very small depth multilinear arithmetic formulas – constitute rather strong proof systems that are strictly stronger than PC, PCR and resolution. Moreover, such multilinear proof systems are capable of refuting efficiently (negations of) families of tautologies that were found hard[2] for other proof systems, such as the bounded-depth Frege proof system. Furthermore, we illustrate a link between multilinear proofs and multilinear arithmetic circuit lower bounds.

1.2.1.1 Background and Motivations

There is a great amount of literature devoted to proving lower bounds on the maximal *degree* of polynomials appearing in PC refutations of some set of initial polynomials (cf. Razborov (1998), Impagliazzo et al. (1999), Buss et al. (2001), Ben-Sasson and Impagliazzo (1999), Alekhnovich et al. (2004), Alekhnovich and Razborov (2001), Razborov (2002-2003)). These lower bounds imply a lower bound on the *size* of the refutations only when polynomials are represented as *a sum of monomials*, that is, as depth-2 arithmetic formulas. For instance, Impagliazzo et al. (1999) showed that any degree lower bound that is linear in the number of variables implies an exponential lower bound on the number of monomials in the refutation.

With respect to lower bounds on the refutation size of algebraic proof systems other than PC and PCR (in which the size of refutations is measured by the number of monomials appearing in the refutations), not much is known. Moreover, extending the PC proof system by allowing it to manipulate general (i.e., not necessarily multilinear) arithmetic formulas makes this proof system considerably stronger (cf. Buss et al. (1996/97); Pitassi (1997); Grigoriev and Hirsch (2003)). In particular, such an extended PC proof system that manipulates general arithmetic formulas polynomially simulates the entire Frege proof system, which is regarded as a rather strong proof system, and for which no super-polynomial size lower bounds are currently known. Thus, if one seeks to prove size lower bounds on refutation size, it is more reasonable to concentrate on (apparently weaker) extensions of PC (and PCR).

Furthermore, it is well known in proof complexity theory (cf. Beame and Pitassi (1998)) that there is an (informal) correspondence between circuit-based complexity classes and proof systems based on these circuits (that is, proofs in which the proof lines consist of circuits from the prescribed circuit-class). Moreover, super-polynomial size lower bounds on *proofs* manipulating circuits from a given circuit-class were only found after super-polynomial size lower bounds were already proved for *circuits* from the circuit-class itself. Keeping in mind this correspondence, it is important to note that super-polynomial lower bounds on multilinear arithmetic formulas for the determinant

[2]Given a proof system P and a family of tautologies $\{\tau_n \mid n \in \mathbb{N}\}$, we say that τ_n *is hard for* P in case there is no P-proofs of τ_n of size polynomial in the size of τ_n (or equivalently, if there is no polynomial-size P-refutations of $\neg\tau_n$).

and permanent functions, as well as other functions, were recently proved in Raz (2004, 2006) and Aaronson (2004). On the other hand no super-polynomial lower bounds are known for general arithmetic formulas.

In light of the aforesaid, our results show that algebraic proof systems operating with multilinear arithmetic formulas (even of a very small depth) constitute on the one hand fairly strong proof systems extending PC and PCR — and on the other hand, the corresponding circuit-class (i.e., multilinear formulas) does have known super-polynomial lower bounds.

Moreover, as mentioned above, the correspondence between proof systems and circuit-classes is not a formal one, but instead it acts more as a working conjecture in proof complexity theory. Nevertheless, using multilinear proofs we are able to pinpoint an interesting case where this correspondence can be formulated explicitly.

1.2.1.2 Summary of Results and Organization

In Chapter 3 we introduce the multilinear proof systems. We prove three kinds of results. The results of the first kind are polynomial simulations. The second kind of results concerns the problem of proving multilinear arithmetic circuit size lower bounds in connection to multilinear proof systems. The results of the third kind are upper bounds on the refutation size of combinatorial principles that were found hard for other proof systems. Both the simulations and upper bounds results are valid when one restricts the multilinear arithmetic formulas in the refutations to depth at most 3. Specifically, we show the following.

Simulation results: In Chapter 3 (Sections 3.2 and 3.3) we show that Depth-2 fMC polynomially simulates resolution, PC and PCR.

In Chapter 3 (Section 3.3) we provide a general simulation result for multilinear proofs. Specifically, Let S be a sequence of polynomials (not formulas) that forms a PCR proof sequence for some given set Q of multilinear polynomials, and consider the corresponding sequence S' of multilinear polynomials formed by "multilinearization" (see Definition 2.5.2) of the polynomials in S. Then, the general simulation result essentially says that there is an fMC proof of Q of size polynomial in the total size of all the multilinear formulas that compute the polynomials in S'.

Relations with algebraic circuit complexity: In Chapter 3 (Section 3.4) we utilize the general simulation described above to assert the following: proving (an explicit) super-polynomial size separation between algebraic proofs manipulating *general* arithmetic circuits and algebraic proofs manipulating *multilinear* arithmetic circuits implies a super-polynomial size lower bound on multilinear arithmetic circuits for an explicit family of polynomials.

Upper bounds: In Chapter 3 Section 3.5 we demonstrate the following upper bound:

1. Depth-3 fMC has polynomial-size refutations of the Tseitin mod p contradictions (for any p) over fields of characteristic $q \nmid p$ that include a primitive p-th root of unity.

In Chapter 5 we establish more upper bounds on multilinear proofs of hard tautologies. This is accomplished as a consequence of a general simulation result (Corollary 5.2.5) stating that depth-3 multilinear proofs polynomially simulate a certain considerably strong fragment of resolution over linear equations introduced in Chapter 4. Specifically, we show the following:

2. Depth-3 fMC over fields of characteristic 0 has polynomial-size refutations of the pigeonhole principle;

3. Depth-3 fMC has polynomial-size refutations of the Tseitin mod p contradictions (for any p) over any field of characteristic 0.

The short multilinear proofs of Tseitin mod p contradictions established in Chapter 3 (Section 3.5) are *different* from the short proofs of Tseitin mod p demonstrated in Chapter 5. The latter apply to multilinear proofs over any field of characteristic 0, while the former over fields of characteristic $q \nmid p$ that include a primitive p-th root of unity. On the other hand, the former proofs have the advantage of being more direct since they do not rely on small depth-3 representations of the symmetric polynomials (and so it is not known if the latter proofs can be carried out in a "syntactic" multilinear proof system [see the discussion on syntactic versus semantic algebraic proof systems in Section 1.2.3]).

Some consequences. Haken (1985) has shown an exponential lower bound on the size of resolution refutations of the pigeonhole principle. Moreover, exponential lower bounds on the size of resolution refutations of certain Tseitin mod 2 tautologies (that is, Tseitin tautologies based on expanding graphs) are also known (see Urquhart (1987); Ben-Sasson and Wigderson (2001)). We conclude then that depth-3 fMC is exponentially stronger than resolution.

From the known exponential lower bounds on PC and PCR refutation size of certain Tseitin mod p tautologies (cf. Buss et al. (2001); Ben-Sasson and Impagliazzo (1999); Alekhnovich et al. (2004)), we conclude that depth-3 fMC is strictly stronger than PC and PCR.

Note also that Razborov (1998) and subsequently Impagliazzo et al. (1999) have shown an exponential-size lower bound on the size of PC (and PCR) refutations of a low-degree version of the pigeonhole principle. Our depth-3 fMC upper bound is also applicable to this low-degree version (see Section 5.3 for more details on this).

Exponential lower bounds on the size of bounded-depth Frege proofs of the pigeonhole principle were proved in Pitassi et al. (1993) and, independently, in Krajíček et al. (1995). Thus Item (2) above shows an exponential gap of depth-3 fMC over bounded depth Frege for the pigeonhole principle. Similarly, an exponential lower bound on the size of bounded-depth Frege proofs of certain Tseitin mod 2 tautologies was shown in Ben-Sasson (2002). Thus, Items (1) and (3) above implies that also for these Tseitin mod 2 tautologies, depth-3 fMC has an exponential gap over bounded-depth Frege proofs.

The results on multilinear proofs appeared as parts of:

Ran Raz and Iddo Tzameret. **The strength of multilinear proofs**. *Computational Complexity*, 17(3):407–457, 2008.

Ran Raz and Iddo Tzameret. **Resolution over linear equations and multilinear proofs**. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

1.2.2 Resolution over Linear Equations with Applications to Multilinear Proofs

The resolution system is a popular refutation proof system for establishing the unsatisfiability of CNF formulas. Nevertheless, it is well known that resolution cannot provide small, polynomial-size, proofs for many basic counting arguments. The most notable examples of this are the strong exponential lower bounds on the resolution refutation size of the pigeonhole principle and its different variants. Due to the popularity of resolution both in practice, as the core of many automated theorem provers, and as a theoretical case-study in propositional proof complexity, it is natural to consider weak extensions of resolution that can overcome its inefficiency in providing proofs of counting arguments. In Chapter 4 we present proof systems that are extensions of resolution, of various strengths, that are suited for this purpose. In Chapter 5, these proof systems will be furthered linked to multilinear proofs.

The basic proof system we shall study in Chapter 4 is denoted R(lin). The proof-lines in R(lin) proofs are disjunctions of linear equations with integral coefficients over the variables $\vec{x} = \{x_1, \dots, x_n\}$. It turns out that (already proper subsystems of) R(lin) can handle very elegantly basic counting arguments. The following defines the R(lin) proof system. Given an initial CNF, we translate every clause $\bigvee_{i\in I} x_i \vee \bigvee_{j\in J} \neg x_j$ (where I are the indices of variables with positive polarities and J are the indices of variables with negative polarities) pertaining to the CNF, into the disjunction $\bigvee_{i\in I}(x_i = 1) \vee \bigvee_{j\in J}(x_j = 0)$. Let A and B be two disjunctions of linear equations, and let $\vec{a}\cdot\vec{x} = a_0$ and $\vec{b}\cdot\vec{x} = b_0$ be two linear equations (where $\vec{a}, \vec{b}$ are two vectors of n integral coefficients, and $\vec{a}\cdot\vec{x}$ is the scalar product $\sum_{i=1}^{n} a_i x_i$; and similarly for $\vec{b}\cdot\vec{x}$). The rules of inference belonging to R(lin) allow to derive $A \vee B \vee ((\vec{a}-\vec{b})\cdot\vec{x} = a_0 - b_0)$ from $A \vee (\vec{a}\cdot\vec{x} = a_0)$ and $B \vee (\vec{b}\cdot\vec{x} = b_0)$. We can also simplify disjunctions by discarding (unsatisfiable) equations of the form $(0 = k)$, for $k \neq 0$. In addition, for every variable x_i, we shall add an axiom $(x_i = 0) \vee (x_i = 1)$, which forces x_i to take on only Boolean values. A derivation of the empty disjunction (which stands for FALSE) from the (translated) clauses of a CNF is called an *R(lin) refutation* of the given CNF. This way, every unsatisfiable CNF has an R(lin) refutation (this can be proved by a straightforward simulation of resolution by R(lin)).

The basic idea that enables us, in Chapter 5, to connect resolution operating with disjunctions of linear equations and multilinear proofs is this: whenever a disjunction of linear equations is simple enough—and specifically, when it is close to a symmetric function, in a manner made precise—then it can be represented by a small size and small depth multilinear arithmetic formula over large enough fields. We show how to polynomially simulate with multilinear proofs, operating with small depth multilinear formulas, certain short proofs carried inside resolution over linear equations. This enables us to provide polynomial-size multilinear proofs for certain hard tautologies.

More specifically, we introduce a fragment of R(lin), which can be polynomially simulated by depth-3 multilinear proofs (that is, proofs in depth-3 fMC). On the one hand this fragment of resolution over linear equations already is sufficient to formalize in a transparent way basic counting arguments, and so it admits small proofs of the pigeonhole principle and the Tseitin mod p formulas (which yields upper bounds on multilinear proofs); and on the other hand we can use the (monotone) interpolation technique to establish an exponential-size lower bound on refutations in this fragment as well as demonstrating a general (non-monotone) polynomial upper bound on interpolants for this fragment.

In Chapter 4 (Section 4.7) we consider the relation of the *cutting planes* system and its extensions with the R(lin) proof system. The cutting planes proof system operates with linear *inequalities* with integral coefficients, and this system is very close to the extensions of resolution we will study in Chapter 4. In particular, the following simple observation can be used to polynomially simulate cutting planes proofs with polynomially bounded coefficients (and some of its extensions) inside resolution over linear equations: the truth value of a linear inequality $\vec{a}\cdot\vec{x} \geq a_0$ (where $\vec{a}$ is a vector of n integral coefficients and $\vec{x}$ is a vector of n *Boolean* variables) is equivalent to the truth value of the following disjunction of linear equalities:

$$(\vec{a}\cdot\vec{x} = a_0) \vee (\vec{a}\cdot\vec{x} = a_0 + 1) \vee \cdots \vee (\vec{a}\cdot\vec{x} = a_0 + k)\ ,$$

where $a_0 + k$ equals the sum of all positive coefficients in $\vec{a}$ (that is, $a_0 + k = \max_{\vec{x}\in\{0,1\}^n}(\vec{a}\cdot\vec{x})$).

1.2.2.1 Comparison to Earlier Work

To the best of our knowledge our results are the first that consider the complexity of resolution proofs operating with disjunctions of linear *equations*. Previous works considered extensions of resolution over linear *inequalities* augmented with the cutting planes inference rules (the resulting proof system denoted R(CP)). In full generality, we show that resolution over linear equations can polynomially simulate R(CP) when the coefficients in all the inequalities are polynomially bounded (however, the converse is not known to hold). On the other hand, we shall consider a certain fragment of resolution over linear equations, in which we do not even know how to polynomially simulate cutting planes proofs with polynomially bounded coefficients in inequalities (let alone R(CP) with polynomially bounded coefficients in inequalities). We now discuss the previous work on R(CP) and related proof systems.

Extensions of resolution to disjunctions of linear *inequalities* were first considered in Krajíček (1998) who developed the proof systems LK(CP) and R(CP). The LK(CP) system is a first-order (Gentzen-style) sequent calculus that operates with linear inequalities instead of atomic formulas and augments the standard first-order sequent calculus inference rules with the cutting planes inference rules. The R(CP) proof system is essentially resolution over linear inequalities, that is, resolution that operates with disjunctions of linear inequalities instead of clauses.

The main motivation of Krajíček (1998) was to extend the feasible interpolation technique and consequently the lower bounds results, from cutting planes and resolution to stronger proof systems. That paper establishes an exponential-size lower bound on a restricted version of R(CP) proofs, namely, when the number of inequalities in each proof-line is $O(n^{\varepsilon})$, where n is the number of variables of the initial formulas, ε is a small enough constant and the coefficients in the cutting planes inequalities are polynomially bounded.

Other papers considering extensions of resolution over linear inequalities are the more recent papers by Hirsch and Kojevnikov (2006) and Kojevnikov (2007). The first paper Hirsch and Kojevnikov (2006) considers combinations of resolution with LP (an incomplete subsystem of cutting planes based on simple linear programming reasoning), with the 'lift and project' proof system (L&P), and with the cutting planes proof system. That paper also illustrates polynomial-size refutations of the Tseitin mod 2 tautologies in all these extensions of resolution. The second paper Kojevnikov (2007) deals with improving the parameters of the tree-like R(CP) lower-bounds obtained in Krajíček (1998). Also, on the more practical level, Hirsch, Itsykson, Kojevnikov, Kulikov, and Nikolenko (2005) have developed an experimental SAT-solver (that is, a software tool for deciding satisfiability) named *basolver*, which stands for mixed Boolean-Algebraic Solver. This SAT-solver solves CNF formulas (and also checks Boolean circuits for equivalence) by translating them first into systems of polynomial equations and disjunctions of *polynomial* equations, and then solving these systems by means of derivation rules in the spirit of the resolution derivation rules.

Whereas previous results concerned primarily with extending the cutting planes proof system, our foremost motivation is to link resolution over linear equations to multilinear proofs. As mentioned above, motivated by relations with multilinear proofs operating with depth-3 multilinear formulas, we shall consider a certain subsystem of resolution over linear equations. For this subsystem we apply twice the interpolation by a communication game technique. The first application is of the *non*-monotone version of the technique, and the second application is of the *monotone* version. Namely, the first application provides a general (non-monotone) interpolation theorem that demonstrates a polynomial (in the size of refutations) upper bound on interpolants; The proof uses the general method of transforming a refutation into a Karchmer-Wigderson communication game for two players, from which a Boolean circuit is then attainable. In particular, we shall apply the interpolation theorem of Krajíček (1997). The second application of the (monotone) interpolation by a communication game technique is implicit and proceeds by using the lower bound criterion of Bonet, Pitassi, and Raz (1997). This criterion states that (semantic) proof systems (of a certain natural and standard kind) whose proof-lines (considered as Boolean functions) have low communication complexity cannot prove efficiently a certain tautology (namely, the clique-coloring tautologies).

1.2.2.2 Summary of Results and Organization

The proof systems. In Chapter 4 we formally define two extensions of resolution of decreasing strength allowing resolution to operate with disjunctions of linear equations. The size of a linear equation $a_1x_1 + \ldots + a_nx_n = a_0$ is in fact $\Sigma_{j=0}^{n} |a_j|$, where $|a_j|$ is the size of the coefficients a_j written in *unary notation*. The size of a disjunction of linear equations is the total size of all linear equations in the disjunction. The size of a proof operating with disjunctions of linear equations is the total size of all the disjunctions in it.

R(lin): This is the stronger proof system (described above) that operates with disjunctions of linear equations with integer coefficients.

R^0(lin): This is a (provably proper) fragment of R(lin). It operates with disjunctions of (arbitrarily many) linear equations whose variables have constant coefficients, under the restriction that every disjunction can be partitioned into a constant number of sub-disjunctions, where each sub-disjunction either consists of linear equations that differ only in their free-terms or is a (translation of a) clause.[3] For the precise definition see Section 4.1.3.

Note that any single linear *inequality* with Boolean variables can be represented by a disjunction of linear equations that differ only in their free-terms (see the example mentioned above in this section). So the R^0(lin) proof system is close to a proof system operating with disjunctions of constant number of linear inequalities (with constant integral coefficients). In fact, disjunctions of linear equations varying only in their free-terms, have more (expressive) strength than a single inequality. For instance, the PARITY function can be easily represented by a disjunction of linear equations, while it cannot be represented by a single linear inequality (nor by a polynomial-size disjunction of linear inequalities).

As already mentioned, the motivation to consider the restricted proof system R^0(lin) comes from its relation to multilinear proofs operating with depth-3 multilinear formulas: R^0(lin) corresponds roughly to the subsystem of R(lin) that we know how to simulate by depth-3 multilinear proofs (the technique is based on converting disjunctions of linear forms into symmetric polynomials, which are known to have small depth-3 multilinear formulas). This simulation is then applied in order to obtain upper bounds on depth-3 multilinear proofs, as R^0(lin) is already sufficient to efficiently prove certain "hard tautologies". Moreover, we are able to establish an exponential lower bound on R^0(lin) refutations size (see below for both upper and lower bounds on R^0(lin) proofs). We also establish a super-polynomial separation of R(lin) from R^0(lin) (via the clique-coloring principle, for a certain choice of parameters; see below).

Upper bounds. In Chapter 4 we demonstrate the following short refutations in R^0(lin) and R(lin):

1. Polynomial-size refutations of the pigeonhole principle in R^0(lin);

[3]The *free-term* of a linear form $a_1x_1 + \ldots + a_nx_n + a_0$ is a_0.

2. Polynomial-size refutations of Tseitin mod p graph formulas in $\mathrm{R}^0(\mathrm{lin})$;

3. Polynomial-size refutations of the clique-coloring formulas in R(lin) (for certain parameters). The refutations here follow by direct simulation of the Res(2) refutations of clique-coloring formulas from Atserias et al. (2002).

Interpolation results. In Chapter 4 Section 4.5 we provide a polynomial upper-bound on (non-monotone) interpolants corresponding to $\mathrm{R}^0(\mathrm{lin})$ refutations; Namely, we show that any $\mathrm{R}^0(\mathrm{lin})$-refutation of a given formula can be transformed into a (non-monotone) Boolean circuit computing the corresponding interpolant function of the formula (if there exists such a function), with at most a polynomial increase in size. We employ the general interpolation theorem for semantic proof systems from Krajíček (1997).

Lower bounds. In Chapter 4 Section 4.6 we provide the following exponential lower bound:

Theorem 1.2.1 $\mathrm{R}^0(\mathrm{lin})$ *does not have sub-exponential refutations for the clique-coloring formulas.*

This result is proved by applying a result of Bonet, Pitassi, and Raz (1997). The result in Bonet et al. (1997) (implicitly) use the monotone interpolation by a communication game technique for establishing an exponential-size lower bound on refutations of general semantic proof systems operating with proof-lines of low communication complexity.

Relations with cutting planes proofs. A proof system combining resolution with cutting planes was presented in Krajíček (1998). The resulting system is denoted R(CP) (see Section 4.7 for a definition). When the coefficients in the linear inequalities inside R(CP) proofs are polynomially bounded, the resulting proof system is denoted R(CP*). In Chapter 4 Section 4.7 we establish the following simulation result:

Theorem 1.2.2 R(lin) *polynomially simulates resolution over cutting planes inequalities with polynomially bounded coefficients* R(CP*).

We do not know whether the converse simulation holds.

From resolution over linear equations to multilinear proofs. Chapter 5 is devoted to establish the following theorem, connecting multilinear proofs with resolution over linear equations:

Theorem 1.2.3 *Multilinear proofs operating with depth-*3 *multilinear formulas over characteristic* 0 *polynomially-simulate* $\mathrm{R}^0(\mathrm{lin})$.

An immediate corollary of this theorem and the upper bounds in $\mathrm{R}^0(\mathrm{lin})$ described in Chapter 4 are the polynomial-size multilinear proofs for the pigeonhole principle and the Tseitin mod p formulas.

The results on resolution over linear equations appeared as parts of:

Ran Raz and Iddo Tzameret. **Resolution over linear equations and multilinear proofs**. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

1.2.3 Symbolic proofs of polynomial identities: From Semantic to Syntactic Algebraic Proofs

Let $\mathbb{F}$ be a field (say, the complex numbers) and let Φ be an arithmetic formula in the input variables $x_1, \ldots, x_n$, computing a polynomial in the ring of polynomials $\mathbb{F}[x_1, \ldots, x_n]$. A *symbolic operation* is any transformation of a subformula in Φ into another subformula, by means of the standard polynomial-ring axioms (expressing associativity and commutativity of both addition and multiplication, distributivity of multiplication over addition, equalities involving only field elements and the laws for the 0 and 1 elements in the field).

Chapter 6 deals with the following basic question:

How many symbolic operations does one need to perform on Φ in order to validate that Φ computes the zero polynomial?

To this end we define the notion of *symbolic proofs of polynomial identities* as follows: assume that the arithmetic formula Φ computes the zero polynomial, then a *symbolic proof* of this fact is a sequence of arithmetic formulas, where the first formula is Φ, the last formula is the formula 0, and every formula in the sequence (excluding the first one) is derived from the (immediate) previous formula in the sequence by a symbolic operation. We are interested in the number of proof-lines in such proof sequences.

1.2.3.1 Background

The problem of deciding whether a given arithmetic circuit (or formula) over some field computes the zero polynomial – namely, the *polynomial identity testing* problem (PIT, for short) – is of great importance in algebraic complexity theory, and complexity theory in general. It is known that when the underlying field is big enough there is an efficient *probabilistic* procedure for deciding whether an arithmetic circuit computes the zero polynomial (cf. Schwartz (1980); Zippel (1979)). However, not much is known about the complexity of *deterministic* algorithms for this problem. Devising an efficient deterministic procedure, or even demonstrating (non-deterministic) sub-exponential witnesses, for the polynomial identity testing problem is a fundamental open problem.

The importance and apparent difficulty in finding an efficient deterministic procedure (or sub-exponential non-deterministic witnesses for that matter) for PIT led researchers to several different directions. On the one hand, there is a growing body of work dedicated to establishing efficient deterministic procedures for PIT when arithmetic circuits are replaced by more restrictive models of computing polynomials (cf. Raz and Shpilka (2005); Dvir and Shpilka (2006); Kayal and Saxena (2007); Karnin and Shpilka (2008); Shpilka and Volkovich (2008)). On the other hand, in a somewhat more logical vein, evidence or "justifications" for the apparent empirical difficulty in finding efficient deterministic algorithms for PIT were discovered in Kabanets and Impagliazzo (2004) (see also, Dvir et al. (2008)).

In Chapter 6 we propose a different direction of research, relevant both to the polynomial identity testing problem as well as to proof complexity. Instead of studying algorithms for the PIT we shall concentrate on symbolic proofs of polynomial identities, that is, proof sequences that manipulate formulas and which have clear and natural structure (besides the fact that they can be efficiently recognized). On the one hand, the choice to study proofs instead of algorithms gives the model more strength (in comparison to algorithms), as one can use non-determinism. On the other hand, we will restrict severely the "reasoning" allowed in these proofs, and this will in turn enable us to demonstrate exponential-size lower bounds on certain proofs of polynomial identities.

1.2.3.2 Motivations

As discussed above, research into the complexity of symbolic proofs of polynomial identities is directed, among others, to achieve better understanding of the polynomial identity testing problem: although it is reasonable to assume (and widely believed) that there *are* polynomial size witnesses (or "proofs") of polynomial identities, lower bounds on certain symbolic proofs of polynomial identities might lead to better understanding of the structure of proofs needed in order to efficiently prove polynomial identities. On the applicative level, our work can be regarded as a contribution to the understanding of the efficiency of symbolic manipulation systems (like symbolic mathematical software tools). Nevertheless, perhaps the most concrete motivation for studying symbolic proofs of polynomial identities comes from algebraic proof systems, as we now explain.

From semantic to syntactic algebraic proof systems. Algebraic propositional proof systems, as discussed before, and as we study in this thesis, are intended to demonstrate that a collection of polynomial equations, derived from the clauses of an unsatisfiable formula has no $0, 1$ solutions over the fixed field.

In previous chapters (Chapters 3, 5) we dealt with *semantic* algebraic propositional proof systems (and not syntactic ones), in which every proof-line is written as an arithmetic formula (or circuit). A semantic algebraic proof is a sequence of arithmetic formulas such that each formula in the sequence computes a formal multivariate polynomial (i.e., an element of the polynomial-ring $\mathbb{F}[x_1, \ldots, x_n]$, where $\mathbb{F}$ is the base field and $x_1, \ldots, x_n$ are the formal variables of the system); note however that for every polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ there is *no unique* arithmetic formula computing p. Thus, each polynomial in the algebraic proof can be written in more than one way as an arithmetic formula. The inference of new polynomials from previous ones, via the algebraic inference rules, is a semantic inference of polynomials from preceding ones, rather than a syntactic inference of formulas from preceding formulas (for instance, the two inference rules mentioned above are semantic in the sense that every root of p in $\mathbb{F}$ is also a root of $q \cdot p$ in $\mathbb{F}$ [for every polynomial q]; and every common root of p and q in $\mathbb{F}$ is also a root of $\alpha \cdot p + \beta \cdot q$, for any $\alpha, \beta \in \mathbb{F}$).

It follows from the aforesaid that semantic algebraic proofs operating with arithmetic formulas might not necessarily be recognizable in polynomial-time (in the sizes of the proofs): because no polynomial-time procedure for the polynomial identity testing problem is known, no known polynomial-time procedure can verify that a proof-line in an algebraic proof was derived correctly from preceding lines in the proof (for instance, the polynomial p might be written as two completely different arithmetic formulas in a proof-line consisting of p and in its legitimate consequence consisting of $q \cdot p$).

Nevertheless, it is sometimes preferable to turn to algebraic proofs that are polynomial-time recognizable. The most natural choice here is to join together the underlying semantic algebraic proof system that operates with arithmetic formulas over a field (similar to that mentioned above), with a symbolic proof system for establishing polynomial identities. This can be achieved, for instance, in the following simple manner: a *syntactic algebraic proof* is defined now to be a sequence of *arithmetic formulas* in which each proof-line is either (i) an initial formula; or (ii) was derived from a previous formula in the sequence by one of the derivation rules pertaining to the symbolic proof system (expressing the polynomial-ring axioms, applied to any subformula); or (iii) was derived by the following inference rules that correspond to the rules of the underlying algebraic proof system: from the *formulas* φ and ψ derive the *formula* $\alpha \times \varphi + \beta \times \psi$ (for α, β field elements); and from the formula φ derive the formula $\psi \times \varphi$ for any arithmetic formula ψ over the base field (the symbol $\times$ stands for the product gate).

Such natural syntactic algebraic propositional proof systems (operating with arithmetic formulas) were mentioned in Buss et al. (1996/97) and were explicitly introduced in Grigoriev and Hirsch (2003). Understanding the complexity of symbolic proofs of polynomial identities is essential in order to understand such syntactic algebraic propositional proof systems operating with arithmetic formulas. Moreover, establishing super-polynomial lower bounds on symbolic proofs of polynomial identities might yield a super-polynomial separation between semantic and syntactic algebraic (propositional) proof systems. For instance, the short (depth-3) multilinear proofs of hard tautologies established in Chapters 5 use in an essential way the fact that the algebraic proof systems are semantic, and it is not known whether such short proofs exist for corresponding syntactic algebraic proof systems.

1.2.3.3 The Basic Model: Analytic Symbolic Proofs

Recall the underlying model of symbolic proofs of polynomial identities illustrated above. We now explain the fragment of symbolic proofs we shall study here. With analogy to traditional research in proof complexity (as well as classical proof theory and automated proofs), we will consider symbolic proof systems that enjoy a property, which is analogous to some extent with the so-called *subformula property* in standard (propositional or predicate sequent calculus) proofs. The subformula property states that every formula that appears in a proof sequence π of T also appears in T. Intuitively, this means that the prover is forbidden from introducing notions not already present in the statement to be proved. Proofs having the subformula property are sometimes called *analytic* (or *cut-free* in the framework of the sequent calculus), and we shall adopt this term here.

Accordingly, we will introduce a proof system for the set of arithmetic formulas computing the zero polynomial, called *analytic symbolic proofs*, in which the following (relaxed form of the) subformula property holds: if π is a proof sequence intended to establish that the formula Φ computes the zero polynomial, then every subformula that appears in some proof-line in π is "originated" from the initial formula to be proved. More formally, this means that for every proof-line and every monomial (with its coefficient) that is syntactically computed in the proof-line, the same monomial is also syntactically computed in the initial proof-line (see Section 6.2 for more details on this and for the definition of syntactic computation of monomials).

The analytic criterion thus implies, for instance, that one cannot add arbitrary formulas computing the zero polynomial in the proof (for example, one cannot get from the proof-line φ to the proof-line $\varphi + f - f$, where f is some arbitrarily chosen arithmetic formula).

The (analytic) proof system we introduce is a natural proof system since, first, symbolic manipulations of polynomial formulas according to the polynomial-ring axioms is something familiar to every high-school student, and second, the restriction to analytic proofs forbids only "ingenious" steps as illustrated above (that is, adding a formula $f - f$, and then using in some clever way this f to efficiently derive the formula 0 in the system).

1.2.3.4 Results

The main technical result in Chapter 6 is an exponential-size lower bound on analytic symbolic proofs of certain hard formulas computing the zero polynomial, where proofs operate with depth-3 formulas and conform to a certain regularity condition on the structure of proofs.

The hard formulas we provide are based on small depth-3 formulas for the elementary symmetric polynomials. We establish a lower bound rate of $2^{\Omega(\sqrt{\ell})}$, where ℓ is the number of variables in the initial hard formulas.

The regularity condition intends to keep the following requirement: once a proof-line $A \times (B + C)$ is transformed into the proof-line $A \times B + A \times C$, in no way the two formulas $A \times B$ and $A \times C$, as well as any other two formulas that originate (among others) from $A \times B$ and $A \times C$ (in a manner made precise), be united together again into a product formula by means of the distributivity rule. For instance, in our case, after $A \times (B + C)$ was broken into the two sums $A \times B + A \times C$, these two sums ($A \times B$ and $A \times C$) cannot be united together again into a product formula by means of the "backward" distributivity rule, to yield $A \times (B + C)$, once more.

Our lower bound follows by a structural analysis of symbolic proofs, and specifically, by tracing the "paths" in which monomials and subformulas "move" along the proof.

Some basic algebraic properties of the small depth-3 formulas of the elementary symmetric polynomials are also exploited in the lower bound argument.

The results in this chapter appear in:

> Iddo Tzameret. **On the Structure and Complexity of Symbolic Proofs of Polynomial Identities**. *Manuscript*, 35 pages, April 2008.[4]

1.2.4 Propositional Proofs under a Promise

Any standard sound and complete propositional proof system has the ability to separate the set of unsatisfiable formulas in conjunctive normal form (CNF) from the set of CNF formulas having at least one satisfying assignment, in the sense that every unsatisfiable CNF has a refutation in the system, while no satisfiable CNF has one. In Chapter 7 we develop and study, within the framework of propositional proof complexity, systems that are "sound and complete" in a relaxed sense: they can separate the set of unsatisfiable CNF formulas from the set of CNF formulas having *sufficiently many* satisfying assignments (where the term "sufficiently many" stands for an explicitly given function of the number of variables in the CNF). We call such proof systems *promise refutation systems*, as they are complete and sound for the set of CNF formulas promised to be either unsatisfiable or to have many satisfying assignments.

Our first task in Chapter 7 is to introduce a natural model for promise propositional refutation systems. This is accomplished by augmenting standard resolution (or any other propositional proof system extending resolution) with an additional collection of axioms, the *promise axioms*. Each refutation in a promise refutation system can make use of at most one promise axiom. The promise axioms are meant to capture the idea that we can ignore or "discard" a certain number of truth assignments from the space of all truth assignments, and still be able to certify (due to the promise) whether or not the given CNF is unsatisfiable. The number of assignments that a promise axiom is allowed to discard depends on the promise we are given, and, specifically, it needs to be less than the number of assignments promised to satisfy a given CNF (unless it is unsatisfiable).

Assuming we have a promise that a satisfiable CNF has more than Λ satisfying assignments, we can discard up to Λ assignments. We refer to Λ as the *promise*. This way, the refutation system is guaranteed not to contain refutations of CNF formulas having more than Λ satisfying assignments, as even after discarding (at most Λ) assignments, we still have at least one satisfying assignment left. On the other hand, any unsatisfiable CNF formula has a refutation in the system, as resolution already has a refutation of it.

We now explain what it means to "discard" assignments and how promise axioms formulate the notion of discarding the *correct number* of truth assignments. Essentially, we say that a truth assignment $\vec{a}$ is *discarded* by some Boolean formula if $\vec{a}$ falsifies the formula. More formally, let $X := \{x_1, ..., x_n\}$ be the set of underlying variables of a given CNF, called the *original variables*. Let A be some CNF formula in the variables

[4]Independently of this paper, Pavel Hrubeš recently investigated close problems concerning equational systems (Hrubeš (2008)); the two papers are about to be substantially expanded and merged.

X, and assume that A also contains variables not from X, called *extension variables*. Let $\vec{a} \in \{0,1\}^n$ be a truth assignment for the X variables, and assume that there is no extension of $\vec{a}$ (assigning values to the extension variables) that satisfies A. Thus, any assignment satisfying A must also satisfy $X \not\equiv \vec{a}$ (that is, $A \models X \not\equiv \vec{a}$), and so any (implicationally) complete proof system can prove $X \not\equiv \vec{a}$ from A, or, in the case of a refutation system, can refute $X \equiv \vec{a}$, given A. In this case, we say that the assignment $\vec{a}$ is *discarded by* A.

The promise axioms we present enjoy two main properties:

1. They discard assignments from the space of possible assignments to the variables X.

2. They express the fact that not too many assignments to the variables X are being discarded (in a manner made precise).

The first property is achieved as follows: let C be any Boolean circuit with n output bits. Then we can formulate a CNF formula A (using extension variables) expressing the statement that the output of C is (equal to) the vector of variables X. This enables A to *discard every truth assignment to the variables of X that is outside the image of the Boolean map defined by C*, because, if an assignment $\vec{a}$ to X is not in the image of C, then no extension of $\vec{a}$ can satisfy A—assuming the formulation of A is correct. (For technical reasons, the actual definition is a bit different from what is described here; see Section 7.2.)

The second property is achieved as follows: assume we can make explicit the statement that the *domain* of the map defined by the Boolean circuit C above is of size at least $2^n - \Lambda$. (See Section 7.2 in Chapter 7 for details on this.) Then, for the second property to hold, it is sufficient that the axiom formulates the statement that the circuit C defines an *injective* map (and thus the image of the map contains enough truth assignments), which can be done quite naturally.

Given a certain promise and its associated promise axiom, we call a refutation of resolution, augmented with the promise axiom, a *resolution refutation under the (given) promise*.

Besides introducing the model of promise refutation systems, our second task in Chapter 7 will be to investigate the basic properties of this model and in particular to determine its average-case proof complexity with respect to different sizes of promises (see below for a summary of our findings in this respect).

1.2.4.1 Background and Motivation

A natural relaxation of the problem of unsatisfiability certification is to require that, if a CNF is satisfiable, then it actually has many satisfying assignments. As mentioned above, we call the specific number of assignments (as a function of the number of variables n) required to satisfy a satisfiable CNF formula, the “promise”. It is thus natural to ask whether giving such a promise can help in obtaining shorter proofs of unsatisfiability.

It is possible to define an *abstract promise proof system* in an analogous manner to the definition of an abstract (or formal) proof system (Definition 1.1.1). Nevertheless, as we already discussed in previous sections, proof complexity theory usually deals with specific and structured proof systems. In accordance with this, we shall be interested in Chapter 7 not with abstract proof systems (that is, not with finding general witnesses for unsatisfiability, possibly under a promise), but rather with specific and more structured proof systems, and specifically with refutation systems built up as extensions of resolution.

In the case of a *big* promise, that is, a constant fraction of the space of all truth assignments ($\Lambda = \varepsilon \cdot 2^n$, for a constant $0 < \varepsilon < 1$), there is already a *deterministic polynomial-time algorithm* for any fixed natural number k that certifies the unsatisfiability of all unsatisfiable kCNF formulas under the promise: the algorithm receives a kCNF that is either unsatisfiable or has more than Λ satisfying assignments and answers whether the formula is unsatisfiable (in case the formula is satisfiable the algorithm provides a satisfying assignment). See Hirsch (1998); Trevisan (2004) for such efficient algorithms.[5] This trivially implies the existence of polynomial-size witnesses for any unsatisfiable kCNF under the promise $\varepsilon \cdot 2^n$. But does resolution already admit such short witnesses of unsatisfiability (that is, resolution refutations) under a big promise? We show that the answer is positive (for all unsatisfiable 3CNF formulas).

In the case of a *smaller* promise, by which we mean $\Lambda = 2^{\delta n}$ for a constant $0 < \delta < 1$, it is possible to efficiently transform any CNF over n variables to a new CNF with $n' = \lceil n/(1-\delta) \rceil$ variables, such that the original CNF is satisfiable if and only if the new CNF has at least $2^{\delta n'}$ satisfying assignments.[6] Thus, the *worst-case* complexity of certifying CNF unsatisfiability under such a promise is polynomially equivalent to the worst-case complexity of certifying CNF unsatisfiability without a promise. However, it is still possible that a promise of $2^{\delta n}$ might give some advantage (that is, a super-polynomial speedup over refutations without a promise) in certifying the unsatisfiability of certain (but not all) CNF formulas; for instance, in the *average-case*.[7]

Feige, Kim, and Ofek (2006) have shown that when the number of clauses is $\Omega(n^{7/5})$ there exist polynomial-size witnesses for the unsatisfiability of 3CNF formulas in the *average-case*. On the other hand, Beame, Karp, Pitassi, and Saks (2002) and Ben-Sasson and Wigderson (2001) showed that resolution does not provide sub-exponential refutations for 3CNF formulas in the average-case when the number of clauses is at most $n^{(3/2)-\epsilon}$, for any constant $0 < \epsilon < 1/2$. [8] This shows that general witnessing of 3CNF unsatisfiability

[5]In the case the promise is $\Lambda = 2^n/poly(n)$, the algorithm in Hirsch (1998) also gives a deterministic sub-exponential time procedure for unsatisfiability certification of kCNF formulas (for a constant k).

[6]This can be achieved simply by adding new $(n' - n)$ "dummy variables". For instance, by adding the clauses of a tautological CNF in these dummy variables to the original CNF. This way, if the original CNF has at least one satisfying assignment then the new CNF has at least $2^{n'-n} \geq 2^{\delta n'}$ satisfying assignments.

[7]Note that if we add dummy variables to a 3CNF then we obtain an "atypical instance" of a 3CNF. Thus, assuming we have polynomial-size witnesses of unsatisfiability of 3CNF formulas under a small promise in the average-case (that is, the "typical case"), the reduction alone (that is, adding dummy variables) does *not* automatically yield polynomial-size witnesses for 3CNF formulas in the average-case without a promise as well.

[8]Beame et al. (2002) showed such a lower bound for $n^{(5/4)-\epsilon}$ number of clauses (for any constant

is strictly stronger than resolution refutations. But is it possible that, under a promise of $2^{\delta n}$, resolution can do better in the average-case? We show that the answer is negative.

There are two main motivations for studying propositional proofs under a given promise and their complexity. The first is to answer the natural question whether CNF unsatisfiability certification enjoys any advantage given a certain promise. As already mentioned, the answer is positive when the promise is a constant fraction of all the truth assignments, and our results imply that this phenomenon already occurs for resolution. For a small promise of $2^{\delta n}$, we can show that, at least in the case of resolution refutations of most 3CNF formulas (of certain clause-to-variable density), the answer is negative. In fact, we can show that the answer stays negative even when the promise is bigger than $2^{\delta n}$, and specifically when $\Lambda = 2^n/2^{n^\xi}$ for some constant $0 < \xi < 1$. Overall, our results establish the first unsatisfiability certification model in which a promise of a certain given size is known to help (that is, allow more efficient certifications) in the average-case, while promises of smaller sizes do not help.

The second motivation is more intrinsic to proof complexity theory: it is a general goal to develop natural frameworks for propositional proofs that are not sound in the strict sense, but rather possess an approximate notion of soundness (like showing that certain "approximations" give speed-ups). For this purpose, the proof systems we propose formalize—in a natural way—the notion of separating unsatisfiable CNF formulas from those that have many satisfying assignments. The promise axioms we present also allow for a natural way of controlling the size of the promise, which in addition leads to an exponential separation between different size promises.

Chapter 7 develops the notion of propositional proofs under a promise, analyzes the proof complexity of these proof systems, as well as obtaining a separation, with respect to different promise sizes, and also illustrates several new facts about the resolution proof system.

1.2.4.2 Results and Organization

After defining the notion of *proofs under a promise* in Chapter 7 Section 7.2, we show that resolution refutations are already enough to efficiently separate unsatisfiable 3CNF formulas from those 3CNF formulas with an arbitrarily small constant fraction of satisfying assignments. In particular, in Section 7.3, we show the following:

Upper Bound: *Let* $0 < \varepsilon < 1$ *be any constant and let* $\Lambda = \varepsilon \cdot 2^n$ *be the given promise. Then* every *unsatisfiable 3CNF with* n *variables has a polynomial-size (in* n*) resolution refutation under the promise* Λ.

The proof of this resembles a deterministic algorithm of Trevisan (2004) for approximating the number of satisfying assignments of kCNF formulas.

$0 < \epsilon < 1/4$). Ben-Sasson and Wigderson (2001) introduced the size-width tradeoff that enables one to prove an exponential lower bound for random 3CNF formulas with $n^{(3/2)-\epsilon}$ number of clauses (for any constant $0 < \epsilon < 1/2$), but the actual proof for this specific clause-number appears in Ben-Sasson (2001).

In contrast to the case of a big promise, our results show that, at least for resolution, a small promise of $\Lambda = 2^{\delta n}$ (for any constant $0 < \delta < 1$) does not give any advantage over standard resolution (that is, resolution without the promise axioms) in most cases (that is, in the average-case). Specifically, in Section 7.4 we show the following:

Lower Bound: *Let $0 < \delta < 1$ be any constant and let $\Lambda = 2^{\delta n}$ be the given promise. Then, there is an exponential lower bound on the size of resolution refutations of random 3CNF formulas under the promise Λ, when the number of clauses is $O(n^{3/2-\epsilon})$, for $0 < \epsilon < \frac{1}{2}$.*

This lower bound actually applies to a more general model of promise proofs. It remains valid even if we allow (somehow) the promise proofs to discard *arbitrarily chosen* sets of truth assignments (of size $\Lambda = 2^{\delta n}$), and not necessarily those sets that are definable by (small) Boolean circuits. In fact, the lower bound applies even to a bigger promise of $\Lambda = 2^{n-n^{\xi}}$, for some constant $0 < \xi < 1$.

The proof strategy for this lower bound follows that of Ben-Sasson and Wigderson (2001) (the *size-width tradeoff* approach), and so the rate of the lower bound matches the one in that paper. The main novel observation is that under the appropriate modifications this strategy also works when one restricts the set of all truth assignments to a smaller set (that is, from 2^n down to $2^n - 2^{\delta n}$ for a constant $0 < \delta < 1$, and in fact down to $2^n - 2^n/2^{n^{\xi}}$, for some constant $0 < \xi < 1$).

It is important to note that these two main results show that the decision to discard sets of truth assignments defined by *Boolean circuits* does not affect the results in any way, and thus should not be regarded as a restriction of the model of promise refutations (at least not for resolution). To see this, note that we could allow a promise refutation to discard *arbitrarily chosen* sets of truth assignments (of the appropriate size determined by the given promise), that is, sets of truth assignments that are not necessarily definable by (small) Boolean circuits. However, although this modification strengthens the model, it is not really necessary for the *upper bound*, as this upper bound is already valid when one discards sets of truth assignments by (small) Boolean circuits. On the other hand, as mentioned above, the *lower bound* is already valid when one allows a promise refutation to discard any *arbitrarily chosen* set of truth assignments (of the appropriate size).

The exact model of promise propositional proof systems is developed in Section 7.2.

The results in this chapter appeared in:

Nachum Dershowitz and Iddo Tzameret. **Complexity of propositional proofs under a promise**. *Proceedings of the Thirty-Fourth International Colloquium on Automata, Languages and Programming (ICALP)*, pages 291–302, 2007. Also in arXiv:0202.7057. (Full journal version submitted.)

Chapter 2

Preliminaries

2.1 General Notations

For a natural number m, we use $[m]$ to denote $\{1, \ldots, m\}$. For a vector of n (integral) coefficients $\vec{a}$ and a vector of n variables $\vec{x}$, we denote by $\vec{a} \cdot \vec{x}$ the scalar product $\sum_{i=1}^{n} a_i x_i$. If $\vec{b}$ is another vector (of length n), then $\vec{a} + \vec{b}$ denotes the addition of $\vec{a}$ and $\vec{b}$ as vectors, and $c\vec{a}$ (for an integer c) denotes the product of the scalar c with $\vec{a}$ (where, $-\vec{a}$ denotes $-1\vec{a}$). For two linear equations $L_1 : \vec{a} \cdot \vec{x} = a_0$ and $L_2 : \vec{b} \cdot \vec{x} = b_0$, their sum $(\vec{a} + \vec{b}) \cdot \vec{x} = a_0 + b_0$ is denoted $L_1 + L_2$ (and their difference $(\vec{a} - \vec{b}) \cdot \vec{x} = a_0 - b_0$ is denoted $L_1 - L_2$). For two Boolean assignments (identified as $0, 1$ strings) $\alpha, \alpha' \in \{0,1\}^n$ we write $\alpha' \geq \alpha$ if $\alpha'_i \geq \alpha_i$, for all $i \in [n]$ (where α_i, α'_i are the ith bits of α and α', respectively).

2.1.1 CNF Formulas

A CNF formula over the propositional variables $x_1, \ldots, x_n$ is defined as follows: a *literal* is a variable x_i or its negation $\neg x_i$. A *clause* is a disjunction of literals. We treat a clause as a set of literals, that is, we delete multiple occurrences of the same literal in a clause. A *CNF formula* is a conjunction of clauses (sometimes treated also as a *set* of clauses, where the conjunction between these clauses is implicit).

2.2 Resolution and Bounded-Depth Frege Proof Systems

Resolution. Resolution is a complete and sound proof system for unsatisfiable CNF formulas. Let C and D be two clauses containing neither x_i nor $\neg x_i$, the *resolution rule* allows one to derive $C \vee D$ from $C \vee x_i$ and $D \vee \neg x_i$. The clause $C \vee D$ is called the *resolvent* of the clauses $C \vee x_i$ and $D \vee \neg x_i$ on the variable x_i, and we also say that $C \vee x_i$ and $D \vee \neg x_i$ were *resolved over* x_i. The *weakening rule* allows to derive the clause $C \vee D$ from the clause C, for any two clauses C, D.

Definition 2.2.1 (Resolution) *A* resolution proof of the clause *D from a CNF formula K is a sequence of clauses $D_1, D_2, \ldots, D_\ell$, such that: (1) each clause D_j is either a clause of K or a resolvent of two previous clauses in the sequence or derived by the weakening rule from a previous clause in the sequence; (2) the last clause $D_\ell = D$. The* **size** *of a resolution proof is the number of clauses in the proof. A* resolution refutation *of a CNF formula K is a resolution proof of the empty clause $\Box$ from K (the empty clause stands for* FALSE*; that is, the empty clause has no satisfying assignments).*

A proof in resolution (or any of its extensions) is also called a *derivation* or a *proof-sequence*. A proof-sequence containing the proof-lines $D_1, \ldots, D_\ell$ is also said to be a *derivation of* $D_1, \ldots, D_\ell$.

Without loss of generality, we assume that no clause in a resolution refutation contains both x_i and $\neg x_i$ (such a clause is always satisfied and hence it can be removed from the proof).

Bounded-depth Frege. We shall not need an explicit definition for the bounded-depth Frege proof system in this thesis. We only state several exponential gaps between multilinear refutations and bounded depth Frege refutations for specific (families of) CNF formulas (based on known exponential lower bounds on the sizes of bounded-depth Frege refutations of these CNF formulas). For a formal definition of bounded-depth Frege see, e.g., Krajíček (1995) (Definition 4.4.2 there).

A Frege proof system is an implicationally complete proof system (meaning that given any set of propositional formulas T, every formula that is semantically implied from T has a proof from T in the system) whose proof lines consist of formulas over some finite complete set of connectives (a complete set of connectives is one that can represent any Boolean function; usually the connectives $\wedge, \vee, \neg$ are used, augmented with the constant FALSE). A Frege proof system is specified by a finite set of sound and complete inference rules, rules for deriving new propositional formulas from existing ones by (consistent) substitution of formulas for variables in the rules.

A *bounded-depth* Frege proof system is a Frege proof system whose proof lines consist of constant-depth formulas, for some fixed constant (in the case of constant-depth formulas, the connectives $\wedge, \vee$ have unbounded fan-in). As mentioned earlier (Section 1.1.2 in the introduction), we can consider bounded-depth Frege to be a proof system for the set of *unsatisfiable* Boolean formulas, by treating a proof sequence (starting from some initial set of unsatisfiable formulas) that ends with FALSE, as a refutation.

2.3 Basic Notions from Proof Complexity

Recall the definition of a formal (Cook-Reckhow) propositional proof system (or equivalently a formal propositional refutation system) from the introduction (Definition 1.1.1): A *formal propositional proof system* is a polynomial-time algorithm A that receives a Boolean formula F (for instance, a CNF) and a string π over some finite alphabet, such that there exists a π with $A(F, \pi) = 1$ if and only if F is unsatisfiable. Note, for example that resolution is a Cook-Reckhow proof system, since it is complete and sound for the set of unsatisfiable CNF formulas, and given a CNF formula F and a string π it is easy to check in polynomial-time (in F *and* π) whether π constitutes a resolution refutation of F.

We will also consider proof systems that are not necessarily (that is, not known to be) Cook-Reckhow proof systems. Specifically, multilinear proof systems (over large enough fields) meet the requirements of the definition of Cook-Reckhow proof systems, *except* that the condition on A above is relaxed: we allow A to be in *probabilistic* polynomial-time **BPP** (which is not known to be equal to deterministic polynomial-time).

For the basic definitions concerning comparison of proof systems, recall the notions of *simulations* and *separation* (Definition 1.1.2 in the introduction chapter, Section 1.1.3).

2.4 Algebraic Propositional Proof Systems

Algebraic propositional proof systems are proof systems for finite collections of polynomial equations having no $0, 1$ solutions over some fixed field. (Formally, each different field yields a different algebraic proof system.) We will only consider refutations of collections of polynomial equations that are translations of CNF formulas, according to a fixed translation scheme we shall define explicitly (see Definition 3.0.7 for the translation used for PCR and multilinear proofs).

The lines of an algebraic refutation consists of polynomials p_i over the given fixed field. Each such proof-line is interpreted as the polynomial equation $p_i = 0$. If we want to consider the *size* of algebraic refutations we should fix the way polynomials inside refutations are represented.

2.4.0.1 Polynomial Calculus

The Polynomial Calculus is a complete and sound proof system for unsatisfiable CNF formulas (translated to polynomial equations), first considered in Clegg et al. (1996).

Definition 2.4.1 (Polynomial Calculus (PC)). *Let $\mathbb{F}$ be some fixed field and let $Q := \{Q_1, \ldots, Q_m\}$ be a collection of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n]$. Call the set of polynomials $x_i^2 - x_i$, for all variables x_i ($1 \le i \le n$), the set of* Boolean axioms *of* PC.

A PC proof *from Q of a polynomial g is a finite sequence $\pi = (p_1, ..., p_\ell)$ of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n]$ (each polynomial p_i is interpreted as the polynomial equation $p_i = 0$), where $p_\ell = g$ and for each $i \in [\ell]$, either $p_i = Q_j$ for some $j \in [m]$, or p_i is a Boolean axiom, or p_i was deduced from p_j, p_k, where $j, k < i$, by one of the following inference rules:*

Product: *from p deduce $x_i \cdot p$, for some variable x_i ;*

Addition: *from p and q deduce $\alpha \cdot p + \beta \cdot q$, for some $\alpha, \beta \in \mathbb{F}$.*

All the polynomials inside a PC *proof are represented as* sums of monomials. *A* PC refutation of *Q is a proof of 1 (which is interpreted as $1 = 0$) from Q.*

The *size* of a PC proof is defined to be the total number of monomials appearing in the polynomials of the proof. The *degree* of a PC proof is the maximum degree of the polynomials in the proof.

Notice that the Boolean axioms have only $0, 1$ solutions.

Note also that the formal variables of the PC proof system are $x_1, \ldots, x_n$. In order to refute in PC an unsatisfiable CNF formula in the variables $x_1, \ldots, x_n$, we translate a CNF formula into a system of polynomials as follows (again, a polynomial p is interpreted as the polynomial equation $p = 0$). A positive literal x_i translates into $1 - x_i$. A negative literal $\neg x_i$ translates into x_i. A clause, i.e., a disjunction of literals $\ell_1 \vee \ldots \vee \ell_k$, translates into the product of the translations of the literals ℓ_i. A CNF is translated into the set of polynomial translations of its clauses. For example, $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_4)$ translates into the two polynomials $(1 - x_1) \cdot (1 - x_2) \cdot x_3$ and $x_1 \cdot x_4$. It is not hard to see that any assignment of 0,1 (where 0 is interpreted as FALSE and 1 as TRUE) to the variables of a CNF formula F satisfies F if and only if it is a common root of the corresponding system of polynomials, over any given field.

2.4.0.2 Polynomial Calculus with Resolution

The translation of CNF formulas into collections of polynomials, demonstrated in the previous paragraph, makes PC unable to polynomially simulate resolution (Definition 2.2.1). For instance, the clause $\bigvee_{i=1}^{n} x_i$ is translated into the polynomial $\Pi_{i=1}^{n}(1-x_i)$. The number of monomials in $\Pi_{i=1}^{n}(1 - x_i)$ is 2^n, exponential in the number of variables in the clause. For this reason an extension of PC, denoted PCR, that *is* capable of simulating resolution was defined as follows (cf. Alekhnovich et al. (2002)).

Definition 2.4.2 (Polynomial Calculus with Resolution (PCR)). *Let $\mathbb{F}$ be some fixed field and let $Q := \{Q_1, \ldots, Q_m\}$ be a collection of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$. The variables $\bar{x}_1, \ldots, \bar{x}_n$ are treated as new formal variables. Call the set of polynomial equations $x^2 - x$, for $x \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$, plus the polynomial equations $x_i + \bar{x}_i - 1$, for all $1 \le i \le n$, the set of* Boolean axioms of PCR.

The inference rules, proofs *and* refutations *of* PCR *are defined the same as in* PC *[Definition 2.4.1] (except that in* PCR *the polynomials are taken from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$). Similarly to* PC*, all the polynomials in a* PCR *proof are represented as* sum of monomials.

The *size* of a PCR proof is defined to be the total number of monomials appearing in the polynomials of the proof. The *degree* of a PCR proof is the maximum degree of the polynomials in the proof. The *number of steps* of a PCR proof is defined to be the number of polynomials in it (i.e., the length of the proof sequence).

Note that the Boolean axioms of PCR have only $0, 1$ solutions, where $\bar{x}_i = 0$ if $x_i = 1$ and $\bar{x}_i = 1$ if $x_i = 0$.

2.5 Arithmetic and Multilinear Circuits and Formulas

2.5.1 Arithmetic Circuits and Formulas

An *arithmetic circuit* is a directed acyclic graph with unbounded (finite) fan-in and unbounded (finite) fan-out. Every leaf of the graph (i.e., a node of fan-in 0) is labeled with either an input variable or a field element. A field element can also label an edge of the graph. Every other node of the graph is labeled with either $+$ or $\times$ (in the first case the node is a plus gate and in the second case a product gate). We assume that there is only one node of out-degree zero, called the *root*. The *size* of an arithmetic circuit C is the total number of nodes in its graph and is denoted by $|C|$.

An arithmetic circuit computes a polynomial in the ring of polynomials $\mathbb{F}[x_1, \ldots, x_n]$ in the following way. A leaf just computes the input variable or field element that labels it. A field element that labels an edge means that the polynomial computed at its tail (i.e., the node where the edge is directed from) is multiplied by this field element. A plus gate computes the sum of polynomials computed by the tails of all incoming edges. A product gate computes the product of the polynomials computed by the tails of all incoming edges. (Subtraction is obtained using the constant -1.) The output of the circuit is the polynomial computed by the root.

The *depth* of a circuit C is the maximal number of edges in a path from a leaf to the root of C, and is denoted by $\mathrm{dp}(C)$.

Unless otherwise stated, we shall consider all circuits to be *leveled circuits*, that is, circuits where all the nodes of a given level (excluding the bottom level nodes, i.e., the leaves) in the circuit-graph have the *same* labels, and two consequent levels have different labels (i.e., the gates in any path in the circuit alternate between plus and product gates).[1] Any arithmetic circuit with unbounded fan-in gates can be transformed into a leveled circuit that computes the same polynomial, with only a polynomial increase in the size of the circuit. Hence, considering only leveled circuits is not a real restriction here.

We say that a variable x_i *occurs in* an arithmetic circuit if x_i labels one of the leaves of the arithmetic circuit, i.e., x_i is an input variable. We say that an arithmetic circuit has a *plus (product) gate at the root* if the root of the circuit is labeled with a plus (product) gate.

An arithmetic circuit is an *arithmetic formula* if its underlying graph is a tree (with edges directed from the leaves to the root).

Comment: In Chapter 6 it will be more convenient to work with arithmetic formulas as labeled trees with *fan-in at most two* (and where the definition of the depth of formulas is changed accordingly to be the maximal number of alternating blocks of consecutive plus or product gates in a path from the root to a leaf).

2.5.2 Multilinear Polynomials, Circuits and Formulas

A polynomial is *multilinear* if in each of its monomials the power of every input variable is at most one.

Definition 2.5.1 *An arithmetic circuit is* multilinear circuit *(or equivalently,* multilinear arithmetic circuit*) if the polynomial computed by* each *gate of the circuit is multilinear (as a formal polynomial, i.e., as an element of* $\mathbb{F}[x_1, \ldots, x_n]$*). Similarly, an arithmetic formula is* a multilinear formula *(or equivalently,* multilinear arithmetic formula*) if the polynomial computed by* each *gate of the formula is multilinear.*

[1] Chapter 6 will be the exception of this (there we shall consider general formulas with fan-in at most 2).

An additional definition we shall use extensively is the following:

Definition 2.5.2 (Multilinearization operator) *Given a field $\mathbb{F}$ and a polynomial $q \in \mathbb{F}[x_1, \ldots, x_n]$, we denote by $\mathrm{M}[q]$ the unique multilinear polynomial equal to q modulo the ideal generated by all the polynomials $x_i^2 - x_i$, for all variables x_i.*

For example, if $q = x_1^2 x_2 + \alpha x_4^3$ (for some $\alpha \in \mathbb{F}$) then $\mathrm{M}[q] = x_1 x_2 + \alpha x_4$.

2.5.3 Notational Conventions Regarding Arithmetic Formulas

We shall often abuse notation by identifying arithmetic formulas with the polynomials they compute. For instance, if Φ is an arithmetic formula computing the polynomial f, then $\mathrm{M}[\Phi]$ is the multilinear *polynomial* $\mathrm{M}[f]$, and not a formula (note that there can be many arithmetic formulas computing a given polynomial). We can also write, for instance, $\Phi \cdot x_i$ to mean the polynomial $f \cdot x_i$, or $\Phi + x_i$ to mean the polynomial $f + x_i$ (we shall often state explicitly when we refer to the polynomial and not the formula).

Also, given m formulas $\Phi_1, \ldots, \Phi_m$, we usually write $\Phi_1 + \ldots + \Phi_m$ and $\Phi_1 \times \ldots \times \Phi_m$ to designate the *formula* with a plus gate at the root with m children $\Phi_1, \ldots, \Phi_m$, and product gate at the root with m children $\Phi_1, \ldots, \Phi_m$, respectively. When writing a formula like $\Phi_1 \times x_i + \Phi_2$, then the $\times$ gate has precedence over the $+$ gate.

2.5.4 Symmetric Polynomials

A *renaming* of the variables in X is a permutation $\sigma \in S_\ell$ (the symmetric group on $[\ell]$) such that x_i is mapped to $x_{\sigma(i)}$ for every $1 \leq i \leq \ell$.

Definition 2.5.3 (Symmetric polynomial) *Given a set of variables $X = \{x_1, \ldots, x_n\}$, a* symmetric polynomial *f over X is a polynomial in (all the variables of) X such that renaming of variables does not change the polynomial (as a formal polynomial).*

For example, $1 + x_1 + x_2 + x_1 \cdot x_2^3 + x_2 \cdot x_1^3$ is a symmetric polynomial over $X = \{x_1, x_2\}$.

The following theorem is due to M. Ben-Or (cf. Theorem 5.1 in Shpilka and Wigderson (2001)):

Theorem 2.5.4 (Ben-Or) *Let $\mathbb{F}$ be a field of characteristic 0 and let X be a set of ℓ variables $\{x_1, \ldots, x_\ell\}$. For every symmetric multilinear polynomial over X (over the field $\mathbb{F}$) there is a polynomial-size (in ℓ) depth-3 multilinear formula. Moreover, this formula is a leveled multilinear formula with a plus gate at the root.*

Proof. Consider the polynomial

$$\prod_{i=1}^{n} (x_i + t)$$

as a univariate polynomial in t of degree n over $\mathbb{F}[x_1, \dots, x_n]$. Then

$$\prod_{i=1}^{n}(x_i + t) = a_0 + a_1 t + ... + a_n t^n.$$

Note that for every $0 \leq d \leq n$ the coefficient a_{n-d} (of t^{n-d}) is exactly the symmetric multilinear polynomial of degree d:

$$a_{n-d} = \sum_{I \subseteq [n], |I| = d} \prod_{i \in I} x_i.$$

We have

$$\begin{pmatrix} 1 & t & t^2 & \cdots & t^{n-1} & t^n \\ 1 & t & t^2 & \cdots & t^{n-1} & t^n \\ & \ddots & & & & \\ \vdots & & & & & \vdots \\ & & & & \ddots & \\ 1 & t & t^2 & \cdots & t^{n-1} & t^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \prod_{i=1}^{n}(x_i + t) \\ \prod_{i=1}^{n}(x_i + t) \\ \vdots \\ \\ \prod_{i=1}^{n}(x_i + t) \end{pmatrix}$$

Let $b_0, \dots, b_n$ be $n+1$ distinct elements from $\mathbb{F}$. Thus by the previous equation we have:

$$\underbrace{\begin{pmatrix} 1 & b_0 & b_0^2 & \cdots & b_0^{n-1} & b_0^n \\ 1 & b_1 & b_1^2 & \cdots & b_1^{n-1} & b_1^n \\ & \ddots & & & & \\ \vdots & & & & & \vdots \\ & & & & \ddots & \\ 1 & b_n & b_n^2 & \cdots & b_n^{n-1} & b_n^n \end{pmatrix}}_{B} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \prod_{i=1}^{n}(x_i + b_0) \\ \prod_{i=1}^{n}(x_i + b_1) \\ \vdots \\ \\ \prod_{i=1}^{n}(x_i + b_n) \end{pmatrix} \tag{2.1}$$

Consider the matrix B as defined in Equation (2.1). The matrix B is a Vandermonde matrix, and since the b_i's are all distinct elements from $\mathbb{F}$, by basic linear algebra (that is, the fact that the determinant of B is $\prod_{0 \leq j < i \leq n}(b_i - b_j)$), B has an inverse denoted B^{-1}, and so:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = B^{-1} \begin{pmatrix} \prod_{i=1}^{n}(x_i + b_0) \\ \prod_{i=1}^{n}(x_i + b_1) \\ \vdots \\ \\ \prod_{i=1}^{n}(x_i + b_n) \end{pmatrix} \tag{2.2}$$

Now, let $0 \leq d \leq n$ and consider a_d, which is the multilinear symmetric polynomial of degree d. Let $(r_0, \ldots, r_n)$ be the dth row of B^{-1}. Then by (2.2)

$$a_d = r_0 \cdot \prod_{i=1}^{n}(x_i + b_0) + ... + r_n \cdot \prod_{i=1}^{n}(x_i + b_n) = \tag{2.3}$$

$$\sum_{j=0}^{n} r_j \cdot \prod_{i=1}^{n}(x_i + b_j) \tag{2.4}$$

Note that this is indeed a depth-3 multilinear formula with a plus gate at the root. □

Chapter 3

Multilinear Proofs

In this chapter we introduce the notion of multilinear proofs, study some of its basic properties and develop tools to deal with multilinear proofs, as well as provide several separation results. We begin by defining the basic proof system operating with multilinear formulas.

Definition 3.0.5 (Formula Multilinear Calculus (fMC)). *Fix a field $\mathbb{F}$ and let $Q := \{Q_1, \ldots, Q_m\}$ be a collection of multilinear polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$ (the variables $\bar{x}_1, \ldots, \bar{x}_n$ are treated as formal variables). Call the set of polynomials consisting of $x_i + \bar{x}_i - 1$ and $x_i \cdot \bar{x}_i$ for $1 \leq i \leq n$, the* Boolean axioms of fMC.

An fMC proof from Q of a polynomial g *is a finite sequence $\pi = (p_1, ..., p_\ell)$ of multilinear polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$, such that $p_\ell = g$ and for each $i \in [\ell]$, either $p_i = Q_j$ for some $j \in [m]$, or p_i is a Boolean axiom of* fMC*, or p_i was deduced by one of the following inference rules using p_j, p_k for $j, k < i$:*

Product: *from p deduce $q \cdot p$, for some polynomial q in $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$ such that $p \cdot q$ is* multilinear*;*

Addition: *from p, q deduce $\alpha \cdot p + \beta \cdot q$, for some $\alpha, \beta \in \mathbb{F}$.*

All the polynomials in an fMC *proof are represented as multilinear formulas. (A polynomial p_i in an* fMC *proof is interpreted as the polynomial equation $p_i = 0$.) An* fMC refutation of *Q is a proof of 1 (which is interpreted as $1 = 0$) from Q.*

The *size* of an fMC proof π is defined as the total sum of all the formula sizes in π and is denoted by $|\pi|$.

Note that the Boolean axioms have only $0, 1$ solutions, where $\bar{x}_i = 0$ if $x_i = 1$ and $\bar{x}_i = 1$ if $x_i = 0$, for each $1 \leq i \leq n$.

Remark. The product inference rule of fMC in Definition 3.0.5 allows a polynomial p to be multiplied by an arbitrary polynomial q as long as $p \cdot q$ is multilinear. We could have restricted this product rule to allow a polynomial p to be multiplied only by a *variable* from $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ (not occurring already in p). It is not hard to show that fMC refutations with such restricted product rule can polynomially simulate fMC refutations as defined in Definition 3.0.5.

Definition 3.0.6 (Depth-k Formula Multilinear Calculus (depth-k fMC)). *For a natural number k,* depth-k fMC *denotes a restriction of the* fMC *proof system, in which proofs consist of multilinear polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$ represented as multilinear formulas of depth at most k.*

In order to refute an unsatisfiable CNF formula in fMC, we first translate the CNF formula into a system of polynomials via the following translation scheme.

Translation of CNF formulas. In the case of fMC and PCR the polynomial translation of CNF formulas is the following.

Definition 3.0.7 (Polynomial translation of CNF formulas). *The literal x_i translates into $\bar{x}_i$. The literal $\neg x_i$ translates into x_i. A clause, i.e., a disjunction of literals $\ell_1 \vee \ldots \vee \ell_k$, translates into the product of the translations of its literals. A CNF is translated into the set of polynomial translations of its clauses.*

Note that this way the clause $\bigvee_{i=1}^{n} x_i$ translates into $\Pi_{i=1}^{n} \bar{x}_i$, which consists of only one monomial.

It is clear that any assignment of 0,1 values to the variables $x_1, \ldots, x_n$ of a CNF formula F satisfies F if and only if it is a common root of the set of polynomial translations of the clauses of F over the given fixed field (where each variables $\bar{x}_i$ gets the negative value of x_i , i.e., $\bar{x}_i = 0$ if $x_i = 1$ and $\bar{x}_i = 1$ if $x_i = 0$).

Note that this translation scheme yields a set of *multilinear monomials*, since each literal occurs at most once inside a clause. From now on, when considering multilinear proofs or PCR proofs, we shall assume that any CNF formula is translated to a system of polynomials via Definition 3.0.7.

3.0.4.1 Discussion about the fMC Proof System

It is important to clarify the following matter. A proof in fMC, as defined in Definition 3.0.5, is a sequence of formal (multilinear) polynomials, that is, (multilinear) elements of $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$. The *representation* of multilinear polynomials inside an fMC proof sequence is done by *arbitrary multilinear formulas*. Thus, each polynomial in an fMC proof can be represented in more than one way by a multilinear formula (in contrast to PC and PCR proofs, where each polynomial has a unique representation as a sum of monomials [disregarding the order of monomials inside a polynomial]). This means that we can think of the inference of new polynomials from previous ones, via the fMC inference rules, as a *semantic inference* of polynomials from preceding ones, rather than a *syntactic inference* of formulas from preceding formulas (the inference is semantic in the sense that any root of p in $\mathbb{F}$ is also a root of $q \cdot p$ in $\mathbb{F}$; and any common root of p and q in $\mathbb{F}$ is also a root of $\alpha \cdot p + \beta \cdot q$, for any $\alpha, \beta \in \mathbb{F}$).

Accordingly, when we talk about the *size* of an fMC proof (or refutation), we take into account a specific choice of multilinear formulas representing each of the polynomials in the proof sequence (naturally, we shall be interested in the most efficient way to represent each multilinear polynomial by a multilinear formula).

It stems from the aforesaid that fMC is not necessarily a propositional proof system *in the formal sense* (Definition 1.1.1): Since it is an open question whether there exists a polynomial-time algorithm that can decide the identity of two (multilinear) arithmetic formulas, it is open whether there exists a polynomial-time algorithm that can verify the correctness of a given refutation, represented as a sequence of multilinear formulas.

Nevertheless, since there is a *probabilistic* polynomial-time algorithm that can verify the identity of two given arithmetic formulas (cf. Schwartz (1980); Zippel (1979)), any proof of fMC can be recognized in polynomial-time (in the proof size) by a probabilistic algorithm (cf. Pitassi (1997) for some facts about algebraic proof systems over arithmetic circuits and formulas). Also, it is worth noting that for some restricted classes of arithmetic formulas (and circuits) there are known *deterministic* polynomial-time algorithms that decide the identity of any two given arithmetic formulas (and circuits) belonging to the prescribed classes (see, for example, Raz and Shpilka (2005), Dvir and Shpilka (2006), Kayal and Saxena (2007)).

3.1 Basic Manipulations of Arithmetic Formulas

In this section we shall prove simple propositions concerning manipulations of arithmetic formulas in fMC. These propositions will be very useful in the sequel. In particular, we take special care to maintain the small depth of arithmetic formulas inside fMC refutations (this makes the arguments of this section and Section 3.3 a bit tedious).

Notational convention. We say that a polynomial f_1 is *subtracted from* f_2 in an fMC proof, if f_2 is added to $-1 \cdot f_1$ by the addition rule.

Also, recall that a constant from the field (e.g., -1) can label an *edge* in an arithmetic formula, which means that the polynomial computed at the tail of the edge (i.e., the node where the edge is directed from) is multiplied by this constant.

Proposition 3.1.1 *Let Φ be a multilinear formula whose root is a plus gate. Let $x \in \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$ be some variable. Then there exists a multilinear formula $\Phi' := x \times \Phi_1 + \Phi_2$, where x does not occur in Φ_1, Φ_2, such that:*

(i) *Φ' and Φ compute the same polynomial;*

(ii) $|\Phi_1| = 2 \cdot |\Phi|$ *and* $|\Phi_2| = |\Phi|$*;*

(iii) $\mathrm{dp}(\Phi_1) = \mathrm{dp}(\Phi_2) = \mathrm{dp}(\Phi)$*;*

(iv) *the roots of both Φ_1 and Φ_2 are plus gates.*

Proof. Let a be an element of the base field. Denote by $\Phi[a/x]$ the formula that results by substituting each occurrence of x in Φ with a.

Let f be the polynomial computed by Φ. Consider the polynomial f as a polynomial in the variable x only, denoted by $f(x)$ (where now the coefficients of the variable x in $f(x)$ also contain variables). Since f is multilinear, $f(x)$ is of degree 1. Thus, by the Lagrange interpolation formula, the following is equal to the polynomial $f(x)$:

$$x \cdot (f(1) - f(0)) + f(0)$$

(this equality can also be verified in a straightforward manner). Hence, the multilinear formula

$$x \times (\Phi[1/x] - \Phi[0/x]) + \Phi[0/x]$$

computes the polynomial $f(x)$. When considering $f(x)$ as a polynomial in all the variables occurring in Φ, $f(x)$ is precisely the polynomial f.

Therefore, letting $\Phi_1 := \Phi[1/x] - \Phi[0/x]$ and $\Phi_2 := \Phi[0/x]$ concludes the proof (note that Φ_1 is of the same depth as that of Φ, since the root of Φ is a plus gate, and since subtraction can be achieved by labeling the *edge* going out of the root of $\Phi[0/x]$ with -1). (Also notice that if x does not occur in Φ then the proof holds trivially, since $\Phi_1 := \Phi[1/x] - \Phi[0/x] = \Phi - \Phi$, which is a formula computing the zero polynomial.) □

Proposition 3.1.2 *Let Φ be a multilinear formula of depth d, whose root is a plus gate, and let $x_i \in \{x_1, \ldots, x_n\}$ be some variable. Then there is a multilinear formula*

$$\bar{x}_i \times x_i \times \varphi_1 + \bar{x}_i \times \varphi_2 + x_i \times \varphi_3 + \varphi_4$$

that computes the same polynomial as Φ, and such that for all $1 \le j \le 4$:

(i) *φ_j does not contain $x_i, \bar{x}_i$;*

(ii) *φ_j has depth at most d and size $O(|\Phi|)$;*

(iii) *φ_j has a plus gate at the root.*

Proof. We simply apply Proposition 3.1.1 three times. Specifically, by Proposition 3.1.1, there are two depth-d and size $O(|\Phi|)$ multilinear formulas Φ_1, Φ_2 that do not contain $\bar{x}_i$, such that:

$$\Phi = \bar{x}_i \cdot \Phi_1 + \Phi_2 .$$

By Claim 3.1.1 again, there exist four depth-d multilinear formulas $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ that do not contain $x_i, \bar{x}_i$, where each formula is of size $O(|\Phi|)$, and has a plus gate at the root, such that:

$$\begin{aligned} \bar{x}_i \cdot \Phi_1 + \Phi_2 &= \bar{x}_i \cdot (x_i \cdot \varphi_1 + \varphi_2) + \Phi_2 \quad \text{apply Claim 3.1.1 on } \Phi_1, x_i \\ &= \bar{x}_i \cdot x_i \cdot \varphi_1 + \bar{x}_i \cdot \varphi_2 + \Phi_2 \\ &= \bar{x}_i \cdot x_i \cdot \varphi_1 + \bar{x}_i \cdot \varphi_2 + x_i \cdot \varphi_3 + \varphi_4 \quad \text{apply Claim 3.1.1 on } \Phi_2, x_i . \end{aligned}$$

(We treat here all formulas as the polynomials they compute; so the equalities are between polynomials, and not formulas.) □

We need the following claim for the proposition that follows.

Claim: Let Φ_1 be a depth-$d \ge 2$ multilinear formula computing the polynomial f. Let Φ_2 be a multilinear formula for a monomial M (i.e., either a depth-1 multilinear formula having a product gate at the root, or a single variable, or an element of the field). Assume that no variable that occurs in Φ_2 also occurs in Φ_1. Then there is a multilinear formula for $f \cdot M$, with the same gate at the root as that of Φ_1, depth d and size $O(|\Phi_1| \cdot |\Phi_2|)$.

Proof of claim: The claim holds simply by distributivity of multiplication over addition.

If Φ_1 has a product gate at the root then $\Phi_1 \times \Phi_2$ is the desired multilinear formula (note this formula is of depth d).

Assume that Φ_1 has a plus gate at its root.

Recall that we consider all formulas to be leveled. Thus, for some m, there exist m multilinear formulas $\varphi_1, \ldots, \varphi_m$, each either has an (unbounded fan-in) $\times$ gate at the root and depth $\leq d-1$, or has depth 0 (i.e., is an input variable or a field element), such that $\Phi_1 = \varphi_1 + \ldots + \varphi_m$. We can assume without loss of generality that $\mathrm{dp}(\Phi_2) = 1$ (otherwise, we consider Φ_2 to be the formula $\Phi_2 \times 1$). For all $1 \leq i \leq m$, $\mathrm{dp}(\Phi_2 \times \varphi_i) \leq d-1$. Thus, by distributivity of multiplication over addition, the formula

$$\Phi_2 \times \varphi_1 + \ldots + \Phi_2 \times \varphi_m \tag{3.1}$$

computes the polynomial $f \cdot M$ and has size $O(|\Phi_1| \cdot |\Phi_2|)$ and depth d. Since, by assumption, no variable that occurs in Φ_2 also occurs in Φ_1, (3.1) is a *multilinear* formula.
$\blacksquare_{\text{Claim}}$

Proposition 3.1.3 *Let $\Phi = \Phi_1 + \ldots + \Phi_k$ be a multilinear formula of depth d. Let $\varphi_1, \ldots, \varphi_k$ be k formulas, where each φ_i is a multilinear formula of size $\leq s$ for a monomial (i.e., φ_i is either a depth-1 multilinear formula having a product gate at the root, or a single variable, or an element of the field). Denote by f the polynomial computed by $\varphi_1 \times \Phi_1 + \ldots + \varphi_k \times \Phi_k$, and assume that no variable that occurs in φ_i also occurs in Φ_i (for all $1 \leq i \leq k$). Then f has a multilinear formula of size $O(s \cdot |\Phi|)$ and depth $\max\{d, 2\}$.*

Proof. We show that for all $1 \leq i \leq k$, there exists a multilinear formula Φ_i' of size $O(s \cdot |\Phi_i|)$ that computes the polynomial computed by $\Phi_i \times \varphi_i$, and such that one of the following holds:

(i) $\mathrm{dp}(\Phi_i') = \mathrm{dp}(\Phi_i)$ and the gate at the root of Φ_i' is the same as that of Φ_i;

(ii) $\mathrm{dp}(\Phi_i') = 2$ and the root of Φ_i' is a plus gate.

Therefore, the multilinear formula $\Phi_1' + \ldots + \Phi_k'$ computes the polynomial f, and has depth $\max\{d, 2\}$ and size $O(\sum_{i=1}^{k} |\Phi_i| \cdot s) = O(s \cdot |\Phi|)$.

Case 1: Assume that $\mathrm{dp}(\Phi_i) \geq 2$, for some $1 \leq i \leq k$. Then by Claim 3.1, the polynomial computed by $\Phi_i \times \varphi_i$ has a multilinear formula Φ_i' of depth $\mathrm{dp}(\Phi_i)$ and size $O(s \cdot |\Phi_i|)$, with the same gate at the root as Φ_i.

Case 2: Assume that $\mathrm{dp}(\Phi_i) < 2$, for some $1 \leq i \leq k$. Then we can switch to a new formula Φ_i'' that computes the same polynomial as Φ_i, such that $\mathrm{dp}(\Phi_i'') = 2$ and Φ_i'' has a plus gate at the root[1]. Thus, we can apply Case 1 on Φ_i''. $\square$

Proposition 3.1.4 *Let $\Phi_2 + \Phi_1$ be a multilinear formula of depth d, where Φ_2 computes the polynomial f_2 (possibly the zero polynomial) and Φ_1 computes the polynomial f_1. Assume that Φ_1 contains neither the variable x_i nor $\bar{x}_i$. Let $d' := \max\{d, 2\}$. Then the*

[1]If $\Phi_i = a$, for a variable or a field element, a, then switch to $a \times 1 + 0$. If Φ_i has a product gate at the root, then switch to $\Phi_i + 0$. If Φ_i is a sum of variables (and/or field elements) with constant coefficients, $\alpha_1 x_{i_1} + \ldots + \alpha_m x_{i_m}$, then switch to $(\alpha_1 x_{i_1} \times 1) + \ldots + \alpha_m x_{i_m}$.

polynomials $f_2 + f_1 \cdot \bar{x}_i$ and $f_2 + f_1 \cdot (1 - x_i)$ can be proved from one another in depth-d' fMC *proofs of size $O(|\Phi_2| + |\Phi_1|)$.*

Proof. Start from the Boolean axiom $x_i + \bar{x}_i - 1$, and multiply it by f_1 in order to get:

$$(x_i + \bar{x}_i - 1) \cdot f_1 \,. \tag{3.2}$$

If we subtract (3.2) from

$$f_2 + f_1 \cdot \bar{x}_i \,, \tag{3.3}$$

we get

$$f_2 + f_1 \cdot (1 - x_i) \,. \tag{3.4}$$

Similarly, if we add (3.2) to (3.4), we get (3.3).

Open parentheses in (3.2), (3.4) and (3.3), and observe that by Proposition 3.1.3 these three polynomials have all multilinear formulas of depth at most d' and size $O(|\Phi_2|+|\Phi_1|)$. □

3.2 Soundness and Completeness of fMC

We show in this section that fMC is a sound proof system. Then we show a simple completeness proof (for CNF formulas), by demonstrating a depth-2 fMC simulation of resolution. In fact, this simulation also stems from the simulation of PCR by depth-2 fMC (Section 3.3), since PCR simulates resolution.

Proposition 3.2.1 fMC *is a sound proof system. That is, if there exists an* fMC *refutation of a system of multilinear polynomials Q over a field $\mathbb{F}$ then the system of multilinear polynomials Q has no common root in $\mathbb{F}$ with $0, 1$ values.*

Proof. Note that the inference rules of fMC are sound: Any root of p in $\mathbb{F}$ is also a root of $q \cdot p$ in $\mathbb{F}$; and any common root of p and q in $\mathbb{F}$ is also a root of $\alpha \cdot p + \beta \cdot q$ (for any $\alpha, \beta \in \mathbb{F}$).

Let $\pi = (p_1, \ldots, p_\ell)$ be an fMC refutation of Q. Any p_i in π is either a Boolean axiom or a polynomial from Q or p_i was deduced from previous polynomials in π by one of the inference rules.

Then, by the soundness of the inference rules, any common root of the system Q that also satisfies the Boolean axioms in $\mathbb{F}$, is also a root of $p_j \in \pi$, for all $j \leq \ell$ (by induction on the refutation length). Since by definition, the last polynomial in π (i.e., p_ℓ) is 1, then there exists no common root of the system Q and the Boolean axioms in $\mathbb{F}$. This means that Q has no common root in $\mathbb{F}$ with $0, 1$ values. □

We show now that fMC is complete for (polynomial translations of) CNF formulas, when the lines of the refutations consist of depth-2 multilinear formulas. Note that any

CNF formula translates via Definition 3.0.7 into a system of multilinear monomials. In particular, we prove that any resolution refutation of an unsatisfiable CNF can be transformed into a depth-2 fMC refutation of (the polynomial translation of) that CNF, with at most a polynomial increase in size.

Proposition 3.2.2 *Depth-*2 fMC *polynomially simulates resolution. Specifically, if P is a resolution derivation of the clause K, where the size of P (that is, the number of clauses in P) is s, and the number of variables in K is n, then there is a depth-*2 *fMC proof of K with size $O(n \cdot s)$.*

Comment: If we assume that the size s of the resolution refutation P is at least the number of variables n, then the proposition states that there is a depth-2 fMC proof of K with size $O(s^2)$.

Proof. By induction on s. For a given clause D, denote by q_D the polynomial translation of D (via Definition 3.0.7).

Base case: $s = 1$. Then the proof is a single clause D taken from the axioms (the clauses pertaining to the initial CNF formula), which translates into q_D of size $O(n)$.

For the induction step we consider the following three cases. In case D, the last clause of P, is an axiom then q_D is of size $O(n)$. In case D, the last clause of P, was derived by the weakening rule, then $D = A \vee B$ where A is a clause occurring previously in P. By the induction hypothesis we already have the fMC proof of size $O(n \cdot (s-1))$ containing q_A. Thus, by the product rule in fMC we can derive with only a single step the formula computing $q_{A \cdot B} = q_A \cdot q_B$. The formula $q_A \cdot q_B$ has size $O(n)$, and so we are done.

The final case to consider is when the last clause of P is a consequence of an application of the resolution rule. Let C, D be two clauses such that $C \vee D$, the last clause in P, is the resolvent of $C \vee x_i$ and $D \vee \neg x_i$ on the variable x_i. Denote by E the clause containing the common literals of C and D. Thus, there exist two clauses A, B having no common literals such that $C = A \vee E$ and $D = B \vee E$. By definition of the resolution rule, A, B, E do not contain the variable x_i (recall that we assume without loss of generality that no clause in a resolution refutation contains both x_i and $\neg x_i$ and we also delete multiple occurrences of the same literal in a clause, and so C, D contain neither x_i nor $\neg x_i$).

By the induction hypothesis we already have the multilinear monomials $q_{A \vee E \vee x_i} = q_A \cdot q_E \cdot \bar{x}_i$ and $q_{B \vee E \vee \neg x_i} = q_B \cdot q_E \cdot x_i$. We need to derive the monomial $q_{C \vee D} = q_{A \vee B \vee E} = q_A \cdot q_B \cdot q_E$ with an fMC proof of size $O(n)$ (one can clearly extract the precise constant coefficient in the big O notation from the following [constant number] of fMC derivation steps). By Proposition 3.1.4 we can prove from $q_A \cdot q_E \cdot \bar{x}_i$ the multilinear polynomial $q_A \cdot q_E \cdot (1 - x_i)$, with a depth-2 fMC proof of size $O(|q_A \times q_E|) = O(n)$. Now, multiply $q_A \cdot q_E \cdot (1 - x_i)$ by q_B. Since the literals in A, B and E are pairwise disjoint, we get a multilinear polynomial $q_A \cdot q_E \cdot q_B \cdot (1 - x_i)$.

The polynomial $q_B \cdot q_E \cdot x_i$ is multiplied by q_A, which yields the multilinear polynomial $q_A \cdot q_E \cdot q_B \cdot x_i$. Adding $q_A \cdot q_E \cdot q_B \cdot (1 - x_i)$ and $q_A \cdot q_E \cdot q_B \cdot x_i$ we finally arrive at

$q_A \cdot q_E \cdot q_B$. Note that the simulation of the resolution rule application just described is of size $O(n)$.

Also notice that it is possible to represent each arithmetic formula in the simulation with a depth-2 formula (i.e., as a sum of monomials). □

3.3 Simulation Results

In this section we prove a general simulation result for fMC (Theorem 3.3.1). Specifically, we show the following: Let π be a PCR refutation of some initial collection of multilinear polynomials Q over some fixed field. Assume that π has polynomially many steps (i.e., the number of proof lines in the PCR proof sequence is polynomial). If the 'multilinearization' (i.e., the result of applying the $\mathrm{M}[\cdot]$ operator – see Definition 2.5.2) of each of the polynomials in π has a polynomial-size depth-d multilinear formula (with a plus gate at the root), then there is a polynomial-size depth-d fMC refutation of Q. (Note that we only require that the number of steps in π is polynomial. The *size* [i.e., the total number of monomials] of the PCR proof might not be polynomially-bounded.)

A simple consequence of the simulation result is that any PC and PCR refutations of a set of initial multilinear polynomials over some fixed field can be simulated by a depth-2 fMC refutation. Since CNF formulas are translated into sets of multilinear polynomials (via Definition 3.0.7), this shows that with respect to (translations of) CNF formulas, depth-2 fMC is at least as strong as PC and PCR.

Another merit of the simulation result is that it can help in proving upper bounds for fMC refutations. This will be used substantially in Chapter 5.

Theorem 3.3.1 *Fix a field $\mathbb{F}$ and let Q be a set of multilinear polynomials from $\mathbb{F}[x_1,\ldots,x_n,\bar{x}_1,\ldots,\bar{x}_n]$. Let $\pi = (p_1,\ldots,p_m)$ be a* PCR *refutation of Q. For each $p_i \in \pi$, let Φ_i be a multilinear formula for the polynomial $\mathrm{M}[p_i]$. Let S be the total size of all formulas Φ_i, i.e., $S = \Sigma_{i=1}^{m}|\Phi_i|$, and let $d \geq 2$ be the maximal depth of all formulas Φ_i. Assume that the depth of all the formulas Φ_i that have a product gate at the root is at most $d-1$. Then there is a depth-d* fMC *refutation of Q of size polynomial in S.*

Corollary 3.3.2 *Depth-*2 fMC *polynomially simulates* PC *and* PCR.

Proof. Since PCR obviously simulates PC it is sufficient to consider only PCR proofs. Recall that the size of a PCR proof is the total number of monomials in it. Note also that given any multivariate polynomial q, the total number of monomials in q is greater or equal than the total number of monomials in $\mathrm{M}[q]$.

Given a PCR proof $\pi := (p_1,\ldots,p_m)$, represent each multilinear polynomial $\mathrm{M}[p_i]$ as a sum of monomials, and denote this multilinear formula by Φ_i. Each Φ_i is of depth at most 2.

Let $|\pi|$ denote the size of the PCR proof π, i.e., the number of monomials in π, and let $\ell \le n$ be the total number of variables that appear in the polynomials in π. In light of Theorem 3.3.1, we need to show that the total size of all the formulas Φ_i is polynomial in $|\pi|$. Since $|\pi| \ge \ell$, it suffices to show that the total size of all the formulas Φ_i (for $1 \le i \le m$) is $O(\ell \cdot |\pi|)$.

Since each Φ_i is a sum of (multilinear) monomials then the total size of all the formulas Φ_i is just the total size of all the monomials occurring in $\Phi_1, \ldots, \Phi_m$ (ignoring constant factors). Each multilinear monomial in $\Phi_1, \ldots, \Phi_m$ is of size $O(\ell)$. Thus (by the first paragraph of this proof), the total size of all the formulas Φ_i is $O(\ell \cdot |\pi|)$. □

Proof of Theorem 3.3.1. Denote by U the sequence of multilinear polynomials $\mathrm{M}[p_1], \ldots, \mathrm{M}[p_m]$. Suppose that π contains an instance of the PCR product rule: from p_i deduce $x \cdot p_i$, for some $x \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$. Then U contains the polynomials $\mathrm{M}[p_i]$ and $\mathrm{M}[x \cdot p_i]$. Note that $\mathrm{M}[x \cdot p_i]$ does not necessarily equal $x \cdot \mathrm{M}[p_i]$. Thus, an instance of a PCR product rule in π does not necessarily turn into an instance of an fMC product rule in U. This means that the sequence U does not necessarily form a legitimate fMC proof sequence.

Nevertheless, with at most a polynomial increase in size, it is possible to turn U into a depth-d fMC proof. That is, we build from the sequence U a depth-d fMC refutation of Q, denoted π'. The size of π' will be polynomial in the total size of all formulas in U. This is done by adding to U new depth-d fMC proof sequences that simulate all instances of the PCR product rule occurring in π (that is, depth-d fMC proof sequences of $\mathrm{M}[x \cdot p_i]$ from $\mathrm{M}[p_i]$, according to the notations of the previous paragraph).

Claim 3.3 and Lemma 3.3.3 that follow, illustrate how to build a depth-preserving small multilinear proofs of $\mathrm{M}[x \cdot p_i]$ from $\mathrm{M}[p_i]$.

Claim: Let Φ_1 and Φ_2 be two multilinear formulas with a plus gate at the root for the polynomials f_1, f_2, respectively. Assume that both Φ_1 and Φ_2 contain neither x_i nor $\bar{x}_i$. Let $d := \max\{\mathrm{dp}(\Phi_1), \mathrm{dp}(\Phi_2), 2\}$. Then there is a depth-$d$ *fMC* proof of $x_i \cdot f_1 + x_i \cdot f_2$ from $x_i \cdot f_1 + f_2$ with size $O(|\Phi_1| + |\Phi_2|)$.

Proof of claim: Apply the following fMC proof sequence:

1. $x_i \cdot f_1 + f_2$	hypothesis
2. $(1 - \bar{x}_i) \cdot (x_i \cdot f_1 + f_2)$	product of (1)
3. $x_i \cdot \bar{x}_i$	Boolean axiom
4. $(x_i \cdot \bar{x}_i) \cdot f_1$	product of (3)
5. $(1 - \bar{x}_i) \cdot (x_i \cdot f_1 + f_2) + (x_i \cdot \bar{x}_i) \cdot f_1$	(2) plus (4)
6. $x_i + \bar{x}_i - 1$	Boolean axiom
7. $(x_i + \bar{x}_i - 1) \cdot f_2$	product of (6)
8. $(1 - \bar{x}_i) \cdot (x_i \cdot f_1 + f_2) + (x_i \cdot \bar{x}_i) \cdot f_1 + (x_i + \bar{x}_i - 1) \cdot f_2$	(5) plus (7)

Note that the last line is equal to $x_i \cdot f_1 + x_i \cdot f_2$.

We need to make sure that each polynomial in the above proof sequence has a depth-d multilinear formula of size $O(|\Phi_1| + |\Phi_2|)$.

The polynomials in lines 3,6 can obviously be written as constant size depth-1 multilinear formulas.

Considering all other lines in the proof sequence; First open parentheses. We get a sum of constant number of terms, where each term is a product of f_1 or f_2 with a multilinear monomial (or a field element, e.g. -1). For example, line 2 equals: $x_i \cdot f_1 + f_2 - \bar{x}_i \cdot x_i \cdot f_1 - \bar{x}_i \cdot f_2$.

Thus, by Proposition 3.1.3, the polynomials in all the lines of the above proof sequence have depth-d multilinear formulas of size $O(|\Phi_1| + |\Phi_2|)$. $\blacksquare_{\text{Claim}}$

Lemma 3.3.3 *Let p_i be a polynomial from $\mathbb{F}[x_1,\ldots,x_n,\bar{x}_1,\ldots,\bar{x}_n]$, and let $x \in \{x_1,\ldots,x_n,\bar{x}_1,\ldots,\bar{x}_n\}$. Let Φ be a multilinear formula for $\mathbf{M}[p_i]$ having a plus gate at the root and let $d := \max\{\mathrm{dp}(\Phi), 2\}$. Then there is a depth-$d$ fMC proof of $\mathbf{M}[x \cdot p_i]$ from $\mathbf{M}[p_i]$, of size $O(|\Phi|)$.*

Proof. We assume that $x = x_i$ for some $x_i \in \{x_1,\ldots,x_n\}$ (the case of $x \in \{\bar{x}_1,\ldots,\bar{x}_n\}$ is similar).

By Proposition 3.1.2, there are multilinear formulas $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ such that

$$\mathbf{M}[p_i] = \bar{x}_i \cdot x_i \cdot \varphi_1 + \bar{x}_i \cdot \varphi_2 + x_i \cdot \varphi_3 + \varphi_4 \tag{3.5}$$

(the equality here is between polynomials), where for all $1 \le j \le 4$: (i) φ_j does not contain $x_i, \bar{x}_i$, and (ii) φ_j has depth at most d and size $O(|\Phi|)$, and (iii) φ_j has a plus gate at the root.

Multiply the Boolean axiom $\bar{x}_i \cdot x_i$ by φ_1, to get

$$\bar{x}_i \cdot x_i \cdot \varphi_1\,. \tag{3.6}$$

Subtract (3.6) from (3.5). We arrive at the polynomial

$$\bar{x}_i \cdot \varphi_2 + x_i \cdot \varphi_3 + \varphi_4\,. \tag{3.7}$$

By (i,ii,iii) above, both φ_1 and $\varphi_2 + \varphi_3 + \varphi_4$ have depth at most d and size $O(|\Phi|)$ multilinear formulas that do not contain $x_i, \bar{x}_i$. Therefore, by Proposition 3.1.3, both (3.6) and (3.7) have multilinear formulas of depth at most d and size $O(|\Phi|)$.

By Proposition 3.1.4, we can derive from (3.7), with a depth-d fMC proof of size $O(|\Phi|)$,

$$(1 - x_i) \cdot \varphi_2 + x_i \cdot \varphi_3 + \varphi_4\,,$$

which is equal to

$$x_i \cdot (\varphi_3 - \varphi_2) + (\varphi_2 + \varphi_4)\,. \tag{3.8}$$

Since all formulas φ_j have plus gates at their root, then $(\varphi_3 - \varphi_2)$ and $(\varphi_2 + \varphi_4)$ have multilinear formulas of depth $\mathrm{dp}(\Phi)$ and size $O(|\Phi|)$. Thus, by Claim 3.3, there is a depth-d fMC proof of

$$x_i \cdot (\varphi_3 - \varphi_2) + x_i \cdot (\varphi_2 + \varphi_4) = x_i \cdot \varphi_3 + x_i \cdot \varphi_4 \,, \tag{3.9}$$

from (3.8), where the size of the proof is $O(|\Phi|)$.

Now, multiply the Boolean axiom $\bar{x}_i \cdot x_i$ by $(\varphi_1 + \varphi_2)$, and add the result to (3.9). We obtain

$$\bar{x}_i \cdot x_i \cdot \varphi_1 + \bar{x}_i \cdot x_i \cdot \varphi_2 + x_i \cdot \varphi_3 + x_i \cdot \varphi_4 = \mathbf{M}[x_i \cdot p_i] \ . \tag{3.10}$$

Similar to (3.7), polynomial (3.10) can be written as depth-d multilinear formula of size polynomial in $O(|\Phi|)$. □

Concluding the proof of Theorem 3.3.1. Recall that U is the sequence of multilinear formulas $\Phi_1, \ldots, \Phi_m$ (corresponding to the polynomials $\mathbf{M}[p_1], \ldots, \mathbf{M}[p_m]$).

Let p_j, p_k (for $j < k \in [m]$) be some instance of the PCR product rule in π. That is, the polynomial $p_k = x \cdot p_j$ is deduced from p_j, for some $x \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$. We can assume that both Φ_j and Φ_k have plus gates at their root, and so by assumption both have depth at most d (if Φ_ℓ, for $\ell \in \{j,k\}$, has a product gate at the root, then by assumption the depth of the formula is at most $d-1$; hence, we can let Φ_ℓ be the formula $\Phi_\ell + 0$). Thus, by Lemma 3.3.3 there is a depth-d fMC proof of $\mathbf{M}[p_k] = \mathbf{M}[x \cdot p_j]$ from $\mathbf{M}[p_j]$ of size $O(|\Phi_j|)$. We denote this proof (sequence) by S_k. For all instances of the PCR product rule in π, replace the formula Φ_k in U with the proof sequence S_k, excluding the first formula of S_k (note that Φ_j the first formula of S_k, already appears in U). Let π' denote the new sequence of formulas obtained from U by this process.

Now, π' is easily seen to be a depth-d fMC refutation of Q: (i) π' ends with $\mathbf{M}[p_m] = \mathbf{M}[1] = 1$; (ii) Every arithmetic formula in π' is a multilinear formula of depth at most d; and (iii) For every formula Ψ_i in π', computing the polynomial q_i, either Ψ_i was added to π' as a formula in some proof sequence S_j (as defined above), or q_i is the result of applying $\mathbf{M}[\cdot]$ on some polynomial p_ℓ from π (that is, $q_i = \mathbf{M}[p_\ell]$ for some $\ell \in [m]$).

In the first case of (iii), Ψ_i is either an axiom of fMC, or a formula that was deduced by one of fMC's inference rules from preceding formulas in S_j.

In the second case of (iii), p_ℓ is either a (multilinear) polynomial from Q, or a Boolean axiom of PCR, or p_ℓ was deduced by one of the two PCR inference rules from some preceding polynomials in π. If p_ℓ is the Boolean axiom $x_j + \bar{x}_j - 1$ of PCR, for some $j \in [n]$, then $q_i = p_\ell$ is also a Boolean axiom of fMC. If p_ℓ is the Boolean axiom $x_j^2 - x_j$, for some $j \in [n]$, and so $q_i = 0$ (thus, q_i can be discarded from the proof; formally, the zero polynomial can be deduced from any polynomial, by the fMC inference rules).

In case p_ℓ was deduced by the PCR product rule from some preceding polynomial p_k ($k < \ell \in [m]$) in π, then by definition of S_ℓ, q_i stems from preceding polynomials in S_ℓ by an fMC inference rule. Moreover, instances of the PCR addition rule in π are transformed in π' into legal instances of the fMC addition rule, as $\mathrm{M}[\cdot]$ is easily seen to be a linear operator. □

3.4 Separations of Algebraic Proof Systems and Multilinear Circuit Lower Bounds

In this section we use Theorem 3.3.1 to link the separation of certain algebraic proof systems to the problem of proving multilinear arithmetic circuit lower bounds. Specifically, we show that if there is a set of multilinear polynomials Q, for which there exists an explicit polynomial-size refutation manipulating *general* arithmetic circuits (i.e., not necessarily multilinear), then proving a super-polynomial lower bound on the refutation size of Q in a proof system manipulating *multilinear* circuits implies a super-polynomial lower bound on the size of multilinear circuits computing an explicit polynomial (see Section 2.5.1 and Definition 2.5.1 for definitions of arithmetic circuits and multilinear circuits, respectively). In fact, this result can be generalized further (see remark after the proof of Theorem 3.4.4). We shall exploit the fact that we work with semantic algebraic proof systems (like fMC) in which polynomials are represented by arbitrarily chosen arithmetic formulas computing them (this fact enabled us to prove the general simulation result in Theorem 3.3.1).

The following defines algebraic proof systems that manipulate general and multilinear arithmetic circuits:

Definition 3.4.1 (cMC, cPCR).
(i) The cMC *(for Circuit Multilinear Calculus) proof system is identical to* fMC*, except that multilinear polynomials in* cMC *proof sequences are represented by multilinear circuits (instead of multilinear formulas). The* size *of a* cMC *proof π is defined to be the total sum of all the circuit-sizes in π.*
(ii) The cPCR *(for Circuit Polynomial Calculus with Resolution) proof system is identical to* PCR*, except that polynomials in* cPCR *proof sequences are represented by (general) arithmetic circuits (instead of explicit sums of monomials). The* size *of a* cPCR *proof π is defined to be the total sum of all the circuit-sizes in π.*

We now reiterate Theorem 3.3.1 where instead of dealing with depth-d multilinear formulas we deal with multilinear circuits (not necessarily of constant-depth):

Proposition 3.4.2 *Fix a field $\mathbb{F}$ and let Q be an unsatisfiable set of multilinear polynomials from $\mathbb{F}[x_1,\dots,x_n,\bar{x}_1,\dots,\bar{x}_n]$. Let $\pi = (p_1,\dots,p_m)$ be a* PCR *refutation of Q. For each $p_i \in \pi$ let Φ_i be a multilinear circuit for the polynomial $\mathrm{M}[p_i]$ and let S be the total size of all the multilinear circuits Φ_i. Then there is a* cMC *refutation of Q of size polynomial in S.*

Proof. First notice that any manipulation of arithmetic formulas presented in Section 3.1 is also applicable to multilinear circuits. Thus, by a straightforward inspection of the proof of Theorem 3.3.1, one can verify that when substituting the phrases 'depth-d multilinear formulas' by 'multilinear circuits' and 'depth-d fMC' by 'cMC', Theorem 3.3.1 still holds. □

Corollary 3.4.3 *Fix a field $\mathbb{F}$ and let Q be an unsatisfiable set of multilinear polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$. Let $\pi = (p_1, \ldots, p_m)$ be a* PCR *refutation of Q, with a polynomial (in n) number of steps.*[2] *Assume that every* cMC *refutation of Q is of super-polynomial size. Then there exists a polynomial $p_i \in \pi$ such that $\mathbf{M}[p_i]$ has no polynomial-size multilinear circuit.*

Proof. The statement follows immediately from Proposition 3.4.2, as if all polynomials $\mathbf{M}[p_i]$ (for all $1 \leq i \leq m$) had polynomial-size multilinear circuits, there would have been also a polynomial-size cMC refutation of Q, which contradicts the assumption. □

Recall from Definition 1.1.2 that a proof system P_1 has a super-polynomial (exponential, respectively) gap over a proof system P_2 for a family $Q := \{Q_n\}_{n\in\mathbb{N}}$ of unsatisfiable sets of polynomials over a field, if there exist polynomial-size P_1 refutations of Q and every P_2 refutation of Q is of super-polynomial (exponential, respectively) size. In this case we shall also say that Q *super-polynomially (exponentially, respectively) separates P_1 from P_2*. In case we also have *explicit* polynomial-size proofs of Q in P_1 then we shall say that we have an *explicit super-polynomial (exponential, respectively) separation of P_1 from P_2 for Q*. The term *explicit* here means that there is a Turing machine that for any given input-size n (given to the machine in unary representation) outputs the proof of Q_n in P_1 and runs in time polynomial in the size of the proof (similarly, we can speak about an explicit (family of) polynomials).

Theorem 3.4.4 *Fix a field $\mathbb{F}$ and let Q be an unsatisfiable set of multilinear polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$. Assume that there is an explicit super-polynomial (exponential, respectively) size separation of* cPCR *from* cMC *for Q. Then there exists an explicit multilinear polynomial g with no polynomial-size (sub-exponential size, respectively) multilinear circuit.*

Proof. Let $(\Theta_1, \ldots, \Theta_m)$ be the explicit polynomial-size cPCR proof sequence of Q (every Θ_i is an arithmetic circuit). For all $1 \leq i \leq m$ let p_i be the polynomial computed by Θ_i (in other words, we can view $(p_1, \ldots, p_m)$ as a PCR refutation of Q). By assumption m is polynomially bounded by n and any cMC refutation of Q (of any depth) is of super-polynomial size (exponential size, respectively) in n. Thus, by Corollary 3.4.3 there exists an $1 \leq i \leq m$ such that $\mathbf{M}[p_i]$ has no polynomial-size (sub-exponential size, respectively) multilinear circuit.

Let $z_1, \ldots, z_m$ be new variables and consider the polynomial $g := \sum_{j=1}^{m} z_j \cdot \mathbf{M}[p_j]$. Then g has no polynomial-size in n (sub-exponential size in n, respectively) multilinear

[2]Again (as in Section 3.3), we only require that the *number of steps* is polynomial. The *size* of the PCR proof might not be polynomially bounded.

circuit (over $\mathbb{F}$) (as if there was such a multilinear circuit Ψ computing g, we could have obtained a polynomial-size in n (sub-exponential size in n, respectively) multilinear circuit for $\mathrm{M}[p_i]$ by substituting every occurrence of z_j in Ψ, for $j \neq i$, by 0, and every occurrence of z_i in Ψ by 1). (Note that the number of variables in g is polynomially bounded by n, since m is polynomially bounded by n.) □

Remark. Theorem 3.4.4 can be generalized further for any (reasonably defined) pair of arithmetic circuit-classes $\mathcal{C}_1, \mathcal{C}_2$ that are at least as strong as multilinear formulas (that is, when considering $\mathcal{C}_1, \mathcal{C}_2$ instead of arithmetic circuits and multilinear circuits, respectively).

Specifically, denote by $\mathcal{C}_1$PCR the proof system that is similar to PCR, where polynomials are represented by $\mathcal{C}_1$-circuits (instead of explicit sums of monomials); and denote by $\mathcal{C}_2$MC the proof system that is similar to fMC, where multilinear polynomials are represented by $\mathcal{C}_2$-circuits (instead of multilinear formulas). It is not hard to see that if $\mathcal{C}_2$ is any (reasonably defined) arithmetic circuit-class that contains the class of multilinear formulas, then Proposition 3.4.2 is still valid when one considers $\mathcal{C}_2$-circuits and $\mathcal{C}_2$MC refutations instead of multilinear circuits and cMC refutations, respectively.

Thus, the same reasoning that was described above (i.e., in Corollary 3.4.3 and Theorem 3.4.4) implies that if there is an explicit super-polynomial (exponential, respectively) size separation of $\mathcal{C}_1$PCR from $\mathcal{C}_2$MC for some unsatisfiable set of multilinear polynomials Q, then there exists an explicit multilinear polynomial g with no polynomial-size (sub-exponential size, respectively) $\mathcal{C}_2$-circuit.

(In a similar manner, we can speak of $\mathcal{C}_1$PCR versus $\mathcal{C}_2$PCR refutations instead of $\mathcal{C}_1$PCR versus $\mathcal{C}_2$MC refutations. We thus obtain that an explicit super-polynomial (exponential, respectively) size separation of $\mathcal{C}_1$PCR from $\mathcal{C}_2$PCR for some unsatisfiable set of polynomials Q implies the existence of an explicit polynomial g [*not necessarily multilinear*] with no polynomial-size [sub-exponential size, respectively] $\mathcal{C}_2$-circuit.)

3.5 Tseitin's Graph Tautologies

In this section we show an exponential gap of depth-3 fMC over resolution, PC, PCR and bounded-depth Frege for certain Tseitin's graph tautologies. Specifically, we show that for any p the Tseitin mod p formula (see Definition 3.5.1) has a polynomial-size depth-3 fMC refutation over any field of characteristic $q \nmid p$ that includes a primitive p-th root of unity.

Comment: In Chapter 5 (Section 5.3) we will prove the existence of different polynomial-size depth-3 fMC proofs of the Tseitin mod p formulas (these proofs will hold over any field of characteristic 0), as well as polynomial-size depth-3 fMC proofs of the pigeonhole principle.

We shall consider the generalization of the Tseitin's graph tautologies, given in Buss et al. (2001). This generalization can be formulated as a CNF formula, which in turn can be reduced to a more convenient form (we observe that this reduction is efficiently provable in depth-2 fMC). Given this latter form, the refutations of the generalized Tseitin formulas follow in a rather straightforward manner, and we show that such refutations can be done efficiently in depth-3 fMC.

It is worth noting that Grigoriev and Hirsch (2003) have shown a polynomial-size constant depth refutation of the Tseitin mod 2 principle in a formal (i.e., syntactic [see Section 1.1.1]) proof system manipulating general arithmetic formulas of constant-depth (this proof system was denoted $\mathcal{F}$-NS).

Preparatory to the generalized Tseitin principle we start by describing the (original) Tseitin mod 2 principle (cf. Tseitin (1968)). Let $G = (V, E)$ be a connected undirected graph with an *odd* number of vertices n. The Tseitin mod 2 tautology states that there is no sub-graph $G' = (V, E')$, where $E' \subseteq E$, so that for *every* vertex $v \in V$, the number of edges from E' incident to v is odd. This statement is valid, since otherwise, summing the degrees of all the vertices in G' would amount to an odd number (since n is odd), whereas this sum also counts every edge in E' twice, and so is even.

The Tseitin mod 2 principle can be generalized to obtain the Tseitin mod p principle, as was suggested in Buss et al. (2001). Let $p \geq 2$ be some fixed integer and let $G = (V, E)$ be a connected undirected r-regular graph with n vertices and no double edges. Let $G' = (V, E')$ be the corresponding *directed* graph that results from G by replacing every (undirected) edge in G with two opposite directed edges. Assume that $n \equiv 1$ (mod p). Then the Tseitin mod p principle states that there is no way to assign to every edge in E' a value from $\{0, \ldots, p-1\}$, so that:

(i) For every pair of opposite directed edges $e, \bar{e}$ in E', with assigned values a, b, respectively, $a + b \equiv 0$ (mod p); and

(ii) For every vertex v in V, the sum of the values assigned to the edges in E' coming out of v is congruent to 1 (mod p).

The Tseitin mod p principle is valid, since if we sum the values assigned to all edges of E' in pairs we obtain 0 (mod p) (by (i)), where summing them by vertices we arrive at a total value of 1 (mod p) (by (ii) and since $n \equiv 1$ (mod p)).

As a propositional formula (in CNF form) the Tseitin mod p principle is formulated by assigning a variable $x_{e,i}$ for every edge $e \in E'$ and every residue i modulo p. The variable $x_{e,i}$ is an indicator variable for the fact that edge e has an associated value i. The following are the clauses of the Tseitin mod p CNF formula, as translated to polynomials (we call it the Tseitin mod p *formula* to emphasize that it is a translation of a CNF formula). (To be consistent with Buss et al. (2001) we use the notation $\text{BTS}_{G,p}$ which stands for 'Boolean Tseitin mod p'.)

Definition 3.5.1 (Tseitin mod p formula (BTS$_{G,p}$)). *Let $p \geq 2$ be some fixed integer and let $G = (V, E)$ be a connected undirected r-regular graph with n vertices and no double* *edges, and assume that $n \equiv 1 \pmod p$. Let $G' = (V, E')$ be the corresponding directed graph that results from G by replacing every (undirected) edge in G with two opposite directed edges.*

Given a vertex $v \in V$, let the edges in E' coming out of v be $e_{v,1}, \dots, e_{v,r}$ and define the following set of polynomials:

$$\mathrm{MOD}_{p,1}(v) := \left\{ \prod_{k=1}^{r} x_{e_{v,k},i_k} \,\middle|\, i_1, \dots, i_r \in \{0, \dots, p-1\} \text{ and } \sum_{k=1}^{r} i_k \not\equiv 1 \mod p \right\}.$$

The Tseitin mod p *formula, denoted* $\mathrm{BTS}_{G,p}$*, consists of the following multilinear polynomials, where each polynomial is easily seen to be a translation of a clause (via Definition 3.0.7):*

1. $\prod_{i=0}^{p-1} \bar{x}_{e,i}$ *, for all $e \in E'$*
   (expresses that every edge is assigned at least one value from $0, \dots, p-1$);
2. $x_{e,i} \cdot x_{e,j}$ *, for all $i \neq j \in \{0, \dots, p-1\}$ and all $e \in E'$*
   (expresses that every edge is assigned at most one value from $0, \dots, p-1$);
3. $\bar{x}_{e,i} \cdot x_{\bar{e},p-i}$ *and* $x_{e,i} \cdot \bar{x}_{\bar{e},p-i}$,[3]
   for all two opposite directed edges $e, \bar{e} \in E'$ and all $i \in \{0, \dots, p-1\}$
   (expresses condition (i) of the Tseitin mod p principle above);
4. $\mathrm{MOD}_{p,1}(v)$ *, for all $v \in V$*
   (expresses condition (ii) of the Tseitin mod p principle above).

Note that for every edge $e \in E'$, the polynomials of (1,2) in Definition 3.5.1, combined with the Boolean axioms of fMC, force any collection of edge-variables $x_{e,0}, \dots, x_{e,p-1}$ to have exactly one true variable $x_{e,i}$, for some $i \in \{0, \dots, p-1\}$. Also, it is easy to verify that, given a vertex $v \in V$, any assignment σ of $0, 1$ values (to the relevant variables) satisfies both the clauses of (1,2) and the clauses of $\mathrm{MOD}_{p,1}(v)$ if and only if σ corresponds to an assignment of values from $\{0, \dots, p-1\}$ to the edges coming out of v that sums up to $1 \pmod p$.

Definition 3.5.2 *Let $G = (V, E)$ be an undirected graph, and let $\epsilon > 0$. The graph G has* expansion ϵ *if for any subset $S \subseteq V$ of vertices with $|S| \leq |V|/2$, $|N(S)| \geq (1 + \epsilon)|S|$, where $N(S)$ is the set of all vertices from V incident to vertices in S.*

Theorem 3.5.3 **(Buss et al. (2001))** *Let $q \geq 2$ be a prime such that $q \nmid p$ and let $\mathbb{F}$ be a field of characteristic q. Let G be an r-regular graph with n vertices and expansion $\epsilon > 0$. Then, any* PCR*-refutation (over $\mathbb{F}$) of* $\mathrm{BTS}_{G,p}$ *requires degree $\Omega(n)$.*

[3] If $i = 0$ then $x_{\bar{e},p-i}$ and $\bar{x}_{\bar{e},p-i}$ denote $x_{\bar{e},0}$ and $\bar{x}_{\bar{e},0}$, respectively.

It can be proved that there exist *constants* $r, \epsilon > 0$ and an infinite family of r-regular graphs $\{G_i\}_{i=1}^{\infty}$, such that every G_i has ϵ expansion and n_i vertices, and n_i tends to infinity as i tends to infinity (cf. Alon (1986)). Thus, for each G_i pertaining to such a family, the corresponding set $\mathrm{BTS}_{G_i,p}$ contains only linear in n_i many polynomials (since for each vertex v in G_i, $\mathrm{MOD}_{p,1}(v)$ defines $< p^r$ many polynomials). Notice also that every polynomial in $\mathrm{BTS}_{G_i,p}$ is a multilinear monomial and has a constant number of variables. So we conclude that the total number of variables in $\mathrm{BTS}_{G_i,p}$ is linear in n_i and the total number of monomials in $\mathrm{BTS}_{G_i,p}$ is also linear in n_i.

By Theorem 3.5.3, $\mathrm{BTS}_{G_i,p}$ has a linear in n_i *degree* lower bound. By the previous paragraph, this means that $\mathrm{BTS}_{G_i,p}$ has a linear in the total number of variables degree lower bound. By the size-degree tradeoff proved in Impagliazzo et al. (1999), a linear (in the number of variables) lower bound on the degree of PCR refutations implies an exponential (in the number of variables) lower bound on the *size* of PCR refutations (this tradeoff was proved for PC (Corollary 6.3 in Impagliazzo et al. (1999)), but it is also valid for PCR as was observed in Alekhnovich et al. (2002)). Therefore, we have:

Corollary 3.5.4 *Let $q \geq 2$ be a prime such that $q \nmid p$ and let $\mathbb{F}$ be a field of characteristic q. For infinitely many n, there is a graph G with n vertices, such that the Tseitin mod p formula* $\mathrm{BTS}_{G,p}$ *has polynomial-size (i.e., it has polynomially many monomials in n), and any* PCR *refutation over $\mathbb{F}$ of* $\mathrm{BTS}_{G,p}$ *has size exponential in n (i.e., the refutation has exponentially many monomials in n).*

We shall show now that if the field $\mathbb{F}$ in Corollary 3.5.4 contains a primitive p-th root of unity, then for any G there is a *polynomial-size* depth-3 fMC refutation of $\mathrm{BTS}_{G,p}$ (over $\mathbb{F}$). For this purpose, we first transform the Tseitin mod p formula into the following multiplicative version (cf. Buss et al. (2001)):

Definition 3.5.5 (Multiplicative Tseitin mod p ($\mathrm{TS}_{G,p}$)). *Let $\mathbb{F}$ be a field of characteristic $q \nmid p$ having a primitive p-th root of unity, denoted by ω (that is, $\omega \neq 1$ and p is the smallest positive integer such that $\omega^p = 1$). Let $G' = (V, E')$ be the graph corresponding to G as in Definition 3.5.1. Define the abbreviation $y_e := \sum_{i=0}^{p-1} x_{e,i} \cdot \omega^i$ for every edge $e \in E'$. The* multiplicative Tseitin mod p*, denoted* $\mathrm{TS}_{G,p}$*, is the following set of* multilinear *polynomials over* $\mathbb{F}$:

1. $y_e \cdot y_{\bar{e}} - 1$, *for all pairs of opposite directed edges $e, \bar{e} \in E'$* ;
2. $\prod_{j=1}^{r} y_{e_j} - \omega$, *for all $v \in V$, where $e_1, \ldots, e_r$ are the edges coming out of v.*

We emphasize that the formal variables of $\mathrm{TS}_{G,p}$ are $x_{e,i}$ for all $e \in E'$ and $i \in \{0, \ldots, p-1\}$. (In Buss et al. (2001) $\mathrm{TS}_{G,p}$ also included the polynomials $y_e^p - 1$ for all edges $e \in E'$. We shall not need these polynomials for the upper bound.)

Notice that every Boolean assignment to the $x_{e,i}$ variables (where $e \in E'$ and $i \in \{0, \ldots, p-1\}$) that satisfies the polynomials in lines (1,2) and line (3) in $\mathrm{BTS}_{G,p}$, also satisfies the polynomials in line (1) in $\mathrm{TS}_{G,p}$. Indeed, let ρ be a Boolean assignment that satisfies the polynomials in lines (1,2) and line (3) in $\mathrm{BTS}_{G,p}$. Then, by lines (1,2) in $\mathrm{BTS}_{G,p}$ there is exactly one variable $x_{e,i}$ from the variables $x_{e,0}, \ldots, x_{e,p-1}$ in y_e that is set to 1 by ρ, and similarly there is exactly one variable $x_{\bar{e},j}$ from the variables $x_{\bar{e},0}, \ldots, x_{\bar{e},p-1}$ in $y_{\bar{e}}$ that is set to 1 in ρ. Thus, under the assignment ρ, $y_e = \omega^i$ and $y_{\bar{e}} = \omega^j$. By line (3) in $\mathrm{BTS}_{G,p}$ we have that $i + j = 0 \pmod p$, and so $y_e \cdot y_{\bar{e}} = \omega^i \cdot \omega^j = 1$ under the assignment ρ. Similar reasoning shows that every Boolean assignment to the $x_{e,i}$ variables (where $e \in E'$ and $i \in \{0, \ldots, p-1\}$) that satisfies the polynomials in lines (1,2) and line (4) in $\mathrm{BTS}_{G,p}$, also satisfies the polynomials in line (2) in $\mathrm{TS}_{G,p}$.

The previous paragraph shows that $\mathrm{BTS}_{G,p}$ semantically implies $\mathrm{TS}_{G,p}$. In fact, by Buss et al. (2001) there is a PCR-proof of $\mathrm{TS}_{G,p}$ from $\mathrm{BTS}_{G,p}$, such that all the polynomials in the proof are of degree at most pr:

Lemma 3.5.6 (Buss et al. (2001)) *For any r-regular graph G, and for any field $\mathbb{F}$ of characteristic $q \nmid p$ that includes a primitive p-th root of unity, there is a* PCR*-proof (over $\mathbb{F}$) of* $\mathrm{TS}_{G,p}$ *from* $\mathrm{BTS}_{G,p}$*, where the degrees of the polynomials in the proof are at most pr.*[4]

Corollary 3.5.7 *For any r-regular graph G with n vertices, and for any field $\mathbb{F}$ of characteristic $q \nmid p$ that includes a primitive p-th root of unity, there is a depth-*2 fMC *(over $\mathbb{F}$) proof of* $\mathrm{TS}_{G,p}$ *from* $\mathrm{BTS}_{G,p}$ *of size polynomial in n, assuming r is a constant.*

Proof. Since r and p are constants, then by Lemma 3.5.6 there is a PCR-proof of $\mathrm{TS}_{G,p}$ from $\mathrm{BTS}_{G,p}$ of constant-degree. The results of Clegg et al. (1996) imply that any constant-degree PCR-proof can be transformed into a polynomial-size (in the number of variables) PCR proof. The number of (formal) variables in $\mathrm{BTS}_{G,p}$ (and, hence $\mathrm{TS}_{G,p}$) is $2prn$, in other words, polynomial in n. Thus, there is a PCR-proof of $\mathrm{TS}_{G,p}$ from $\mathrm{BTS}_{G,p}$ of size polynomial in n. By Corollary 3.3.2, there is also such a depth-2 fMC proof of size polynomial in n. □

The following is the main theorem of this section:

Theorem 3.5.8 *Let $\mathbb{F}$ be a field of characteristic $q \nmid p$ that includes a primitive p-th root of unity. Let G be an r-regular graph with n vertices. Then, there is a depth-*3 fMC *polynomial-size (in n) refutation of* $\mathrm{BTS}_{G,p}$ *over $\mathbb{F}$.*

Proof. By Corollary 3.5.7, we first derive the polynomials of $\mathrm{TS}_{G,p}$ from $\mathrm{BTS}_{G,p}$, with a depth-2 fMC proof of size polynomial in n.

Given $\mathrm{TS}_{G,p}$, the refutation idea is straightforward: Recall that we interpret a polynomial t in an fMC proof sequence as the equation $t = 0$. Thus, the first axiom of $\mathrm{TS}_{G,p}$ interprets as $y_e \cdot y_{\bar{e}} = 1$, and the second axiom interprets as $\prod_{i=1}^{r} y_{e_i} = \omega$. Therefore, the multiplication of all polynomials $\prod_{i=1}^{r} y_{e_i}$, for all $v \in V$, equals 1, by the first axiom. On

[4]This result was proved in Buss et al. (2001) for PC, when one replaces in $\mathrm{TS}_{G,p}$ and $\mathrm{BTS}_{G,p}$ every occurrence of $\bar{x}_i$ (for any $1 \le i \le n$) by $1 - x_i$. Lemma 3.5.6 clearly stems from this.

the other hand, by the second axiom, this multiplication equals $\omega^n = \omega$ (since $n \equiv 1 \pmod{p}$). So we reached $\omega = 1$, a contradiction.

More formally, the depth-3 fMC refutation goes as follows. For any $v \in V$, denote by E'_v the set of edges from E' that come out of v. Let v_0 be some vertex in V. Apply the following depth-3 fMC proof sequence:

1. $\prod_{e \in E'_{v_0}} y_e - \omega$ hypothesis

2. $\left(\prod_{e \in E'_{v_0}} y_e - \omega\right) \cdot \prod_{v \in V \setminus \{v_0\}} \prod_{e \in E'_v} y_e$

$= \prod_{e \in E'} y_e - \omega \cdot \prod_{v \in V \setminus \{v_0\}} \prod_{e \in E'_v} y_e$ product of (1).

Now, choose a different vertex $v_1 \neq v_0$ from V.

3. $\prod_{e \in E'_{v_1}} y_e - \omega$ hypothesis

4. $\left(\prod_{e \in E'_{v_1}} y_e - \omega\right) \cdot \omega \cdot \prod_{v \in V \setminus \{v_0, v_1\}} \prod_{e \in E'_v} y_e$

$= \omega \cdot \prod_{v \in V \setminus \{v_0\}} \prod_{e \in E'_v} y_e - \omega^2 \cdot \prod_{v \in V \setminus \{v_0, v_1\}} \prod_{e \in E'_v} y_e$ product of (3)

5. $\prod_{e \in E'} y_e - \omega^2 \cdot \prod_{v \in V \setminus \{v_0, v_1\}} \prod_{e \in E'_v} y_e$ add (2) and (4).

Continuing in the same manner for all other vertices $v \in V$, we arrive at $\prod_{e \in E'} y_e - \omega^n$, which equals

6. $\prod_{e \in E'} y_e - \omega\,,$

over $\mathbb{F}$, since $n \equiv 1 \pmod{p}$.

We now substitute by 1 each product $y_e \cdot y_{\bar{e}}$ in (6), for any two opposite directed edges $e, \bar{e} \in E'$. Specifically, choose a pair of opposite directed edges $e_0, \bar{e}_0 \in E'$.

7. $y_{e_0} \cdot y_{\bar{e}_0} - 1$ hypothesis

8. $(y_{e_0} \cdot y_{\bar{e}_0} - 1) \cdot \prod_{e \in E' \setminus \{e_0, \bar{e}_0\}} y_e = \prod_{e \in E'} y_e - \prod_{e \in E' \setminus \{e_0, \bar{e}_0\}} y_e$ product of (7).

In the same manner, let $e_1, \bar{e}_1 \in E'$ be another pair of opposite directed edges. We can multiply $y_{e_1} \cdot y_{\bar{e}_1} - 1$ by $\prod_{e \in E' \setminus \{e_0, \bar{e}_0, e_1, \bar{e}_1\}} y_e$ and reach $\prod_{e \in E' \setminus \{e_0, \bar{e}_0\}} y_e - \prod_{e \in E' \setminus \{e_0, \bar{e}_0, e_1, \bar{e}_1\}} y_e$. Adding this to (8) yields $\prod_{e \in E'} y_e - \prod_{e \in E' \setminus \{e_0, \bar{e}_0, e_1, \bar{e}_1\}} y_e$. Continuing this process for all other pairs of opposite directed edges from E', we arrive finally at

9. $\prod_{e \in E'} y_e - 1$.

Subtracting (9) from (6) we reach $1 - \omega$. Since, $\omega \neq 1$, then $1 - \omega$ has an inverse in the field, so by multiplying $1 - \omega$ by its inverse we arrive at the terminal polynomial 1.

It is easy to verify that when replacing every variable y_e with its corresponding polynomial $\sum_{i=0}^{p-1} x_{e,i} \cdot \omega^i$ (which constitutes a depth-1 multilinear formula with a plus gate at the root), every polynomial in the above proof sequence can be written as a depth-3 multilinear formula of polynomial-size (in n) with a plus gate at the root. □

3.5.1 Consequences: Separation Results

By Corollary 3.5.4 and Theorem 3.5.8, we conclude that:

Corollary 3.5.9 *Over fields of any characteristic q that include a primitive p-th root of unity, where $q \nmid p$, depth-*3 fMC *has an exponential gap over* PC *and* PCR *for Tseitin mod p formulas (when the underlying graphs are appropriately expanding).*

It is known that the Tseitin mod 2 formulas have only exponential-size refutations in resolution (again, when the underlying graphs are appropriately expanding; see Urquhart (1987); Ben-Sasson and Wigderson (2001)). Moreover, Ben-Sasson (2002) proved an exponential lower bound on bounded-depth Frege proof-size of such Tseitin mod 2 formulas. Therefore, by Theorem 3.5.8:

Corollary 3.5.10 *Over fields of characteristic different than* 2 *depth-*3 fMC *has an exponential gap over resolution and bounded-depth Frege for Tseitin mod* 2 *formulas (when the underlying graphs are appropriately expanding).*

Notice that the refutation of the Tseitin mod p formula described in the proof of Theorem 3.5.8 uses only depth-3 multilinear formulas, with a *constant number of product gates* (in other words, the root is a plus gate with a constant fan-in), or depth-2 multilinear formulas (by Corollary 3.5.7). Dvir and Shpilka (2006) (Theorem 6.10) showed a deterministic polynomial-time algorithm for deciding the identity of such formulas – i.e., a polynomial-time algorithm that receives two (leveled) multilinear formulas of depth 3 with a constant fan-in plus gate at the root, and answers whether the two formulas compute the same polynomial (see also Kayal and Saxena (2007)). Thus, depth-3 fMC proof systems for which all depth-3 multilinear formulas appearing in proofs have a constant fan-in plus gate at the root constitute *formal* propositional proof systems (see Section 3.0.4.1) (note that these proof systems can manipulate *any kind* of depth-2 or depth-1 multilinear formulas; we only restrict the way depth-3 multilinear formulas appear). Therefore, by Corollaries 3.5.9 and 3.5.10, we have the following:

Corollary 3.5.11 *Depth-3 fMC proof systems for which all depth-3 multilinear formulas appearing in proofs have a constant fan-in plus gate at the root, are sound and complete* formal *proof systems. Moreover, these formal proof systems are strictly stronger than* PC, PCR *and resolution, and have an exponential gap over bounded-depth Frege (for Tseitin mod 2 formulas, when the underlying field has characteristic different than 2 and the underlying graphs are appropriately expanding).*

3.6 Chapter Summary

In this chapter we introduced a semantic algebraic proof system operating with multilinear formulas. We established basic simulation and separation results and showed that multilinear proofs operating with depth-3 formulas are strictly stronger than resolution, PC and PCR (while depth-2 multilinear proofs already polynomially simulate these three proof systems). The separation results were obtained by demonstrating short proofs for the Tseitin mod p formulas.

In Chapter 5 we will show the existence of short multilinear proofs of depth 3 also for the propositional pigeonhole principle. This will be done by using results established in the next chapter. Specifically, in Chapter 4 we will study fragments of resolution over linear equations. In Chapter 5 we will find out that these fragments can be polynomially simulated by multilinear proofs operating with depth-3 multilinear formulas. Moreover, we will be able to establish an exponential lower bound on such fragments of resolution over linear equations. Nevertheless, the question of lower bounds on multilinear proofs remains open (see discussion in Chapter 8).

Chapter 4

Resolution over Linear Equations

In this chapter we study extensions of resolution in which each proof-line is a disjunction of linear equations with integral coefficients.

4.1 R(lin) and its Fragments

4.1.1 Disjunctions of Linear Equations

For this section we use the convention that all the formal variables in the propositional proof systems considered are taken from the set $X := \{x_1, \ldots, x_n\}$.

For L a linear equation $a_1x_1 + \ldots + a_nx_n = a_0$, the right hand side a_0 is called the *free-term* of L and the left hand side $a_1x_1 + \ldots + a_nx_n$ is called the *linear form* of L (the linear form can be 0). A *disjunction of linear equations* is of the following general form:

$$\left(a_1^{(1)}x_1 + \ldots + a_n^{(1)}x_n = a_0^{(1)}\right) \vee \cdots \vee \left(a_1^{(t)}x_1 + \ldots + a_n^{(t)}x_n = a_0^{(t)}\right) , \tag{4.1}$$

where $t \geq 0$ and the coefficients $a_i^{(j)}$ are integers (for all $0 \leq i \leq n,\ 1 \leq j \leq t$). We discard duplicate linear equations from a disjunction of linear equations. The semantics of such a disjunction is the natural one: we say that an assignment of integral values to the variables $x_1, ..., x_n$ *satisfies* (4.1) if and only if there exists $j \in [t]$ so that the equation $a_1^{(j)}x_1 + \ldots + a_n^{(j)}x_n = a_0^{(j)}$ holds under the given assignment.

The symbol $\models$ denotes the *semantic implication* relation, that is, for every collection $D_1, \ldots, D_m$ of disjunctions of linear equations,

$$D_1, \ldots, D_m \models D_0$$

means that every assignment of $0, 1$ values that satisfies all $D_1, \ldots, D_m$ also satisfies D_0. In this case we also say that $D_1, \ldots, D_m$ *semantically imply* D_0.

The *size of a linear equation* $a_1x_1 + \ldots + a_nx_n = a_0$ is $\sum_{i=0}^{n} |a_i|$, i.e., the sum of the bit sizes of all a_i written in *unary* notation. Accordingly, the *size of the linear form* $a_1x_1 + \ldots + a_nx_n$ is $\sum_{i=1}^{n} |a_i|$. The *size of a disjunction of linear equations* is the total size of all linear equations in it.

Since all linear equations considered in this chapter (and the next one [Chapter 5]) are of integral coefficients, we shall speak of *linear equations* when we actually mean linear equations with integral coefficients. Similar to resolution, the *empty disjunction* is unsatisfiable and stands for the truth value FALSE.

Translation of clauses. We can translate any CNF formula to a collection of disjunctions of linear equations in a direct manner: every clause $\bigvee_{i\in I} x_i \vee \bigvee_{j\in J} \neg x_j$ (where I and J are sets of indices of variables) pertaining to the CNF is translated into the disjunction $\bigvee_{i\in I}(x_i = 1) \vee \bigvee_{j\in J}(x_j = 0)$. For a clause D we denote by $\widetilde{D}$ its translation into a disjunction of linear equations. It is easy to verify that any Boolean assignment to the variables $x_1, \ldots, x_n$ satisfies a clause D if and only if it satisfies $\widetilde{D}$ (where TRUE is treated as 1 and FALSE as 0).

Note on terminology. All the proof systems considered in this chapter intend to prove the *unsatisfiability* over $0, 1$ values of collections of clauses (possibly, of translation of the clauses to disjunctions of linear equations). Therefore, throughout this chapter we shall sometimes speak about refutations and proofs interchangeably, always intending refutations, unless otherwise stated.

4.1.2 Resolution over Linear Equations – R(lin)

Defined below is our basic proof system R(lin) that enables resolution to reason with disjunctions of linear equations. As we wish to reason about Boolean variables we augment the system with the axioms $(x_i = 0) \vee (x_i = 1)$, for all $i \in [n]$, called the *Boolean axioms.*

Definition 4.1.1 (R(lin)) *Let $K := \{K_1, \ldots, K_m\}$ be a collection of disjunctions of linear equations. An* R(lin)-proof from K of a disjunction of linear equations D *is a finite sequence $\pi = (D_1, ..., D_\ell)$ of disjunctions of linear equations, such that $D_\ell = D$ and for every $i \in [\ell]$, either $D_i = K_j$ for some $j \in [m]$, or D_i is a Boolean axiom $(x_h = 0) \vee (x_h = 1)$ for some $h \in [n]$, or D_i was deduced by one of the following* R(lin)*-inference rules, using D_j, D_k for some $j, k < i$:*

Resolution *Let A, B be two, possibly empty, disjunctions of linear equations and let L_1, L_2 be two linear equations. From $A \vee L_1$ and $B \vee L_2$ derive $A \vee B \vee (L_1 - L_2)$.*

Weakening *From a, possibly empty, disjunction of linear equations A derive $A \vee L$, where L is an arbitrary linear equation over X.*

Simplification *From $A \vee (0 = k)$ derive A, where A is a, possibly empty, disjunction of linear equations and $k \neq 0$.*

An R(lin) refutation of *a collection of disjunctions of linear equations K is a proof of the empty disjunction from K.*

Definition 4.1.2 (Size of an R(lin) proof) *The* size *of an* R(lin) *proof π is the total size of all the disjunctions of linear equations in π, denoted $|\pi|$ (where coefficients are written in unary notation).*

Proposition 4.1.3 *From $A \vee L_1$ and $B \vee L_2$ one can derive $A \vee B \vee (L_1 + L_2)$ with only three applications of the resolution rule .*

Proof. From $B \vee L_2$ derive $B \vee (0 = 0)$, by using the resolution rule on $B \vee L_2$ itself. Using once more the resolution rule on $B \vee L_2$ and $B \vee (0 = 0)$ we get $B \vee -L_2$. Apply the resolution rule on $A \vee L_1$ and $B \vee -L_2$, to get $A \vee B \vee (L_1 + L_2)$. $\square$

Similar to resolution, in case $A \vee B \vee (L_1 - L_2)$ is derived from $A \vee L_1$ and $B \vee L_2$ by the resolution rule, we say that $A \vee L_1$ and $B \vee L_2$ were *resolved over L_1 and L_2*, respectively, and we call $A \vee B \vee (L_1 - L_2)$ the *resolvent* of $A \vee L_1$ and $B \vee L_2$. We also describe such an application of the resolution rule by saying that L_1 *was subtracted from* L_2 *in* $A \vee L_1$ *and* $B \vee L_2$. In case $A \vee B \vee (L_1 + L_2)$ is derived from $A \vee L_1$ and $B \vee L_2$ by Proposition 4.1.3, we say that L_1 *was added to* L_2 *in* $A \vee L_1$ *and* $B \vee L_2$.

In light of the direct translation between CNF formulas and collections of disjunctions of linear equations (described in the previous subsection), we can consider R(lin) to be a proof system for the set of unsatisfiable CNF formulas:

Proposition 4.1.4 *The* R(lin) *refutation system is a sound and complete Cook-Reckhow (Definition 1.1.1) refutation system for unsatisfiable CNF formulas (translated into unsatisfiable collection of disjunctions of linear equations).*

Proof. Completeness of R(lin) (for the set of unsatisfiable CNF formulas) stems from a straightforward simulation of resolution, as we now show.

Proceed by induction on the length of the resolution refutation to show that any resolution derivation of a clause A can be translated with only a linear increase in size into an R(lin) derivation of the corresponding disjunction of linear equations $\widetilde{A}$ (see the previous subsection for the definition of $\widetilde{A}$).

The base case: An initial clause A is translated into its corresponding disjunction of linear equations $\widetilde{A}$.

The induction step: If a resolution clause $A \vee B$ was derived by the resolution rule from $A \vee x_i$ and $B \vee \neg x_i$, then in R(lin) we subtract $(x_i = 0)$ from $(x_i = 1)$ in $\widetilde{B} \vee (x_i = 0)$ and $\widetilde{A} \vee (x_i = 1)$, respectively, to obtain $\widetilde{A} \vee \widetilde{B} \vee (0 = 1)$. Then, using the simplification rule, we can cut-off $(0 = 1)$ from $\widetilde{A} \vee \widetilde{B} \vee (0 = 1)$, and arrive at $\widetilde{A} \vee \widetilde{B}$.

If a clause $A \vee B$ was derived in resolution from A by the weakening rule, then we derive $\widetilde{A} \vee \widetilde{B}$ from $\widetilde{A}$ by the weakening rule in R(lin). This concludes the simulation of resolution by R(lin).

Soundness of R(lin) stems from the soundness of the inference rules.

The R(lin) proof system is a Cook-Reckhow proof system, as it is easy to verify in polynomial-time whether an R(lin) proof-line is inferred, by an application of one of R(lin)'s inference rules, from a previous proof-line (or proof-lines). Thus, any sequence of disjunctions of linear equations, can be checked in polynomial-time (in the size of the sequence) to decide whether or not it is a legitimate R(lin) proof-sequence. □

In Section 4.3 we shall see that a stronger notion of completeness (that is, implicational completeness) holds for R(lin) and its subsystems.

4.1.3 Fragment of Resolution over Linear Equations – R^0(lin)

Here we consider a restriction of R(lin), denoted R^0(lin). As discussed in the introduction (Section 1.2.2) R^0(lin) is roughly the fragment of R(lin) we know how to polynomially simulate with depth-3 multilinear proofs. By results established in the sequel (Sections 4.4.3 and 4.6) R(lin) is *strictly stronger* than R^0(lin), which means that R(lin) polynomially simulates R^0(lin), while the converse does not hold.

R^0(lin) operates with disjunctions of (arbitrarily many) linear equations with constant coefficients (excluding the free terms), under the following restriction: Every disjunction can be partitioned into a constant number of sub-disjunctions, where each sub-disjunction either consists of linear equations that differ only in their free-terms or is a (translation of a) clause.

Every linear *inequality* with Boolean variables can be represented by a disjunction of linear equations that differ only in their free-terms. So the R^0(lin) proof system resembles, to some extent, a proof system operating with disjunctions of constant number of linear inequalities with constant integral coefficients (on the other hand, it is probable that R^0(lin) is stronger than such a proof system, as a disjunction of linear equations that differ only in their free terms is [expressively] stronger than a linear inequality: the former can define the PARITY function while the latter cannot).

Example of an R^0(lin) *proof-line:*

$$(x_1 + \ldots + x_\ell = 1) \vee \cdots \vee (x_1 + \ldots + x_\ell = \ell) \vee (x_{\ell+1} = 1) \vee \cdots \vee (x_n = 1),$$

for some $1 \leq \ell \leq n$. Note that in the left part $(x_1 + \ldots + x_\ell = 1) \vee \cdots \vee (x_1 + \ldots + x_\ell = \ell)$ every disjunct has the same linear form [excluding the free term] with coefficients $0, 1$, while the right part $(x_{\ell+1} = 1) \vee \cdots \vee (x_n = 1)$ is a translation of a clause. The next section contains other concrete (and natural) examples of R^0(lin) proof-lines.

Let us define formally the R^0(lin) proof system. To this end, we introduce the following definition.

Definition 4.1.5 ($\mathbf{R}_{\mathbf{c},\mathbf{d}}$(lin)-line) *Let D be a disjunction of linear equations whose variables have integer coefficients with absolute values at most c (the free-terms are unbounded). Assume D can be partitioned into at most d sub-disjunctions $D_1, \ldots, D_d$, where each D_i either consists of (an unbounded) disjunction of linear equations that differ only in their free-terms, or is a translation of a clause (as defined in Subsection 4.1.1). Then the disjunction D is called an* $\mathrm{R}_{c,d}$(lin)-line.

The size *of an* $\mathrm{R}_{c,d}$(lin)*-line D is defined as before, that is, as the total size of all equations in D, where coefficients are written in unary notation.*

Thus, any $\mathrm{R}_{c,d}$(lin)-line is of the following general form:

$$\bigvee_{i \in I_1} \left(\vec{a}^{(1)} \cdot \vec{x} = \ell_i^{(1)} \right) \vee \cdots \vee \bigvee_{i \in I_k} \left(\vec{a}^{(k)} \cdot \vec{x} = \ell_i^{(k)} \right) \vee \bigvee_{j \in J} (x_j = b_j) \,, \tag{4.2}$$

where $k \leq d$ and for all $r \in [n]$ and $t \in [k]$, $a_r^{(t)}$ is an integer such that $|a_r^{(t)}| \leq c$, and $b_j \in \{0, 1\}$ (for all $j \in J$) (and $I_1, \ldots, I_k, J$ are unbounded sets of indices). Clearly, a disjunction of clauses is a clause in itself, and so we can assume that in any $\mathrm{R}_{c,d}$(lin)-line only a *single* (translation of a) clause occurs.

The R^0(lin) proof system is a restriction of R(lin) in which each proof-line is an $\mathrm{R}_{c,d}$(lin)-line, for some apriori chosen constants c, d. More formally, we have the following definition.

Definition 4.1.6 ($\mathbf{R}^0$(lin)) *Let $K := \{K_n \mid n \in \mathbb{N}\}$ be a family of collections of disjunctions of linear equations. Then $\{P_n \mid n \in \mathbb{N}\}$ is a family of* R^0(lin)-proofs *of K if there exist constant integers c, d such that: (i) each P_n is an* R(lin)*-proof of K_n; and (ii) for all n, every proof-line in P_n is an* $\mathrm{R}_{c,d}$(lin)*-line.*

The size *of an* R^0(lin) *proof is defined the same way as the size of* R(lin) *proofs, that is, as the total size of all the proof-lines in the proof (where coefficients are written in unary notation).*

If K_n is a collection of disjunctions of linear equations parameterized by $n \in \mathbb{N}$, we shall say that K_n *has a polynomial-size (in n)* R^0(lin) *proof*, if there are two constants c, d that do not depend on n, and a polynomial p, such that for every n, K_n has R(lin) proof of size at most $p(n)$ in which every proof-line is an $\mathrm{R}_{c,d}$(lin)-line.

The simulation of resolution inside R(lin) (in the proof of Proposition 4.1.4) is carried on with each R(lin) proof-line being in fact a translation of a clause, and hence, an $\mathrm{R}_{1,1}$(lin)-line (notice that the Boolean axioms of R(lin) are $\mathrm{R}_{1,1}$(lin)-lines). This already implies that R^0(lin) is a complete refutation system for the set of unsatisfiable CNF formulas.

4.2 Reasoning and Counting inside R(lin) and its Subsystems

In this section we illustrate a simple way to reason by case-analysis inside R(lin) and its subsystems. This kind of reasoning will simplify the presentation of proofs inside R(lin) (and R^0(lin)) in the sequel. We will then demonstrate efficient and transparent proofs for simple counting arguments that will also facilitate us in the sequel.

4.2.1 Basic Reasoning inside R(lin) and its Subsystems

Given K a collection of disjunctions of linear equations $\{K_1, \dots, K_m\}$ and C a disjunction of linear equations, denote by $K \vee C$ the collection $\{K_1 \vee C, \dots, K_m \vee C\}$. Recall that the formal variables in our proof system are $x_1, \dots, x_n$.

Lemma 4.2.1 *Let K be a collection of disjunctions of linear equations, and let z abbreviate some linear form with integer coefficients. Let $E_1, \dots, E_\ell$ be ℓ disjunctions of linear equations. Assume that for all $i \in [\ell]$ there is an* R(lin) *derivation of E_i from $z = a_i$ and K with size at most s where $a_1, \dots, a_\ell$ are distinct integers. Then, there is an* R(lin) *proof of $\bigvee_{i=1}^{\ell} E_i$ from K and $(z = a_1) \vee \dots \vee (z = a_\ell)$, with size polynomial in s and ℓ.*

Proof. Denote by D the disjunction $(z = a_1) \vee \cdots \vee (z = a_\ell)$ and by π_i the R(lin) proof of E_i from K and $z = a_i$ (with size at most s), for all $i \in [\ell]$. It is easy to verify that for all $i \in [\ell]$ the sequence $\pi_i \vee \bigvee_{j\in[\ell]\setminus\{i\}}(z = a_j)$ is an R(lin) proof of $E_i \vee \bigvee_{j\in[\ell]\setminus\{i\}}(z = a_j)$ from K and D. So overall, given D and K as premises, there is an R(lin) derivation of size polynomial in s and ℓ of the following collection of disjunctions of linear equations:

$$E_1 \vee \bigvee_{j\in[\ell]\setminus\{1\}} (z = a_j), \ldots, E_\ell \vee \bigvee_{j\in[\ell]\setminus\{\ell\}} (z = a_j)\,. \tag{4.3}$$

We now use the resolution rule to cut-off all the equations $(z = a_i)$ inside all the disjunctions in (4.3). Formally, we prove that for every $1 \le k \le \ell$ there is a polynomial-size (in s and ℓ) R(lin) derivation from (4.3) of

$$E_1 \vee \cdots \vee E_k \vee \bigvee_{j\in[\ell]\setminus[k]} (z = a_j)\,, \tag{4.4}$$

and so putting $k = \ell$, will conclude the proof of the lemma.

We proceed by induction on k. The base case for $k = 1$ is immediate (from (4.3)). For the induction case, assume that for some $1 \le k < \ell$ we already have an R(lin) proof of (4.4), with size polynomial in s and ℓ.

Consider the line

$$E_{k+1} \vee \bigvee_{j\in[\ell]\setminus\{k+1\}} (z = a_j)\,. \tag{4.5}$$

We can now cut-off the disjunctions $\bigvee_{j\in[\ell]\setminus[k]}(z = a_j)$ and $\bigvee_{j\in[\ell]\setminus\{k+1\}}(z = a_j)$ from (4.4) and (4.5), respectively, using the resolution rule (since the a_j's in (4.4) and in (4.5) are disjoint). We will demonstrate this derivation in some detail now, in order to exemplify a proof carried inside R(lin). We shall be less formal sometime in the sequel.

Resolve (4.4) with (4.5) over $(z = a_{k+1})$ and $(z = a_1)$, respectively, to obtain

$$(0 = a_1 - a_{k+1}) \vee E_1 \vee \cdots \vee E_k \vee E_{k+1} \vee \bigvee_{j\in[\ell]\setminus\{1,k+1\}} (z = a_j)\,. \tag{4.6}$$

Since $a_1 \neq a_{k+1}$, we can use the simplification rule to cut-off $(0 = a_1 - a_{k+1})$ from (4.6), and we arrive at

$$E_1 \vee \cdots \vee E_k \vee E_{k+1} \vee \bigvee_{j\in[\ell]\setminus\{1,k+1\}} (z = a_j)\,. \tag{4.7}$$

Now, similarly, resolve (4.4) with (4.7) over $(z = a_{k+1})$ and $(z = a_2)$, respectively, and use simplification to obtain

$$E_1 \vee \cdots \vee E_k \vee E_{k+1} \vee \bigvee_{j\in[\ell]\setminus\{1,2,k+1\}} (z = a_j)\,.$$

Continue in a similar manner until you arrive at

$$E_1 \vee \cdots \vee E_k \vee E_{k+1} \vee \bigvee_{j\in[\ell]\setminus\{1,2,\ldots,k,k+1\}} (z = a_j)\,,$$

which is precisely what we need. □

Under the appropriate conditions, Lemma 4.2.1 also holds for $\mathrm{R}^0(\mathrm{lin})$ proofs. Formally we have the following lemma.

Lemma 4.2.2 *Let K be a collection of disjunctions of linear equations, and let z abbreviate a linear form with integer coefficients. Let $E_1, \ldots, E_\ell$ be ℓ disjunctions of linear equations. Assume that for all $i \in [\ell]$ there is an* R(lin) *derivation of E_i from $z = a_i$ and K with size at most s, where the a_i's are distinct integers and where every proof-line is an* $\mathrm{R}_{c,d}(\mathrm{lin})$*-line. Then, assuming $\bigvee_{i=1}^{\ell} E_i$ is an* $\mathrm{R}_{c,d}(\mathrm{lin})$*-line itself, there is an* R(lin) *proof of $\bigvee_{i=1}^{\ell} E_i$ from K and $(z = a_1) \vee \cdots \vee (z = a_\ell)$, with size polynomial in s and ℓ, and with all proof-lines* $\mathrm{R}_{c,d+1}(\mathrm{lin})$*-lines.*

Proof. It can be verified by straightforward inspection that, under the conditions spelled out in the statement of the lemma, each proof-line in the R(lin) derivations demonstrated in the proof of Lemma 4.2.1 is an $\mathrm{R}_{c,d+1}(\mathrm{lin})$-line. □

Abbreviations. Lemmas 4.2.1 and 4.2.2 will sometimes facilitate us to proceed inside R(lin) and $\mathrm{R}^0(\mathrm{lin})$ with a slightly less formal manner. For example, the situation in Lemma 4.2.1 above can be depicted by saying that "if $z = a_i$ implies E_i (with a polynomial-size proof) for all $i \in [\ell]$, then $\bigvee_{i=1}^{\ell}(z = a_i)$ implies $\bigvee_{i=1}^{\ell} E_i$ (with a polynomial-size proof)".

In case $\bigvee_{i=1}^{\ell}(z = a_i)$ above is just the *Boolean axiom* $(x_i = 0) \vee (x_i = 1)$, for some $i \in [n]$, and $x_i = 0$ implies E_0 and $x_i = 1$ implies E_1 (both with polynomial-size proofs), then to simplify the writing we shall sometime not mention the Boolean axiom at all. For example, the latter situation can be depicted by saying that "if $x_i = 0$ implies E_0 with a polynomial-size proof and $x_i = 1$ implies E_1 with a polynomial-size proof, then we can derive $E_0 \vee E_1$ with a polynomial-size proof".

4.2.2 Basic Counting inside R(lin) and R^0(lin)

In this subsection we illustrate how to efficiently prove several basic counting arguments inside R(lin) and $\mathrm{R}^0(\mathrm{lin})$. This will facilitate us in showing short proofs for hard tautologies in the sequel. In accordance with the last paragraph in the previous subsection, we shall carry the proofs inside R(lin) and $\mathrm{R}^0(\mathrm{lin})$ with a slightly less rigor.

Lemma 4.2.3

(i) Let z_1 abbreviate $\vec{a} \cdot \vec{x}$ and z_2 abbreviate $\vec{b} \cdot \vec{x}$. Let D_1 be $\bigvee_{\alpha \in \mathcal{A}}(z_1 = \alpha)$ and let D_2 be $\bigvee_{\beta \in \mathcal{B}}(z_2 = \beta)$, where $\mathcal{A}, \mathcal{B}$ are two (finite) sets of integers. Then there is a polynomial-size (in the size of D_1, D_2) R(lin) *proof from D_1, D_2 of:*

$$\bigvee_{\alpha \in \mathcal{A}, \beta \in \mathcal{B}} (z_1 + z_2 = \alpha + \beta) \ . \tag{4.8}$$

(ii) Moreover, assume that the vector $\vec{a}+\vec{b}$ consists of integers with absolute values at most c (which means that D_1, D_2 are $\mathrm{R}_{c,1}$(lin)*-lines). Then, there is a polynomial-size (in the size of D_1, D_2)* R(lin) *proof of (4.8) from D_1, D_2, with each proof-line being an* $\mathrm{R}_{c,3}$(lin)*-line.*

Proof. Denote the elements of $\mathcal{A}$ by $\alpha_1,\ldots,\alpha_k$. In case $z_1=\alpha_i$, for some $i\in[k]$ then we can add $z_1=\alpha_i$ to every equation in $\bigvee_{\beta\in\mathcal{B}}(z_2=\beta)$ to get $\bigvee_{\beta\in\mathcal{B}}(z_1+z_2=\alpha_i+\beta)$. Therefore, there exist k R(lin) proofs, each with polynomial-size (in $|D_1|$ and $|D_2|$), of

$$\bigvee_{\beta\in\mathcal{B}}(z_1+z_2=\alpha_1+\beta)\,,\ \bigvee_{\beta\in\mathcal{B}}(z_1+z_2=\alpha_2+\beta)\,,\quad\ldots\quad,\bigvee_{\beta\in\mathcal{B}}(z_1+z_2=\alpha_k+\beta)$$

from $z_1=\alpha_1$, $z_1=\alpha_2$,…,$z_1=\alpha_k$, respectively.

Thus, by Lemma 4.2.1, we can derive

$$\bigvee_{\alpha\in\mathcal{A},\beta\in\mathcal{B}}(z_1+z_2=\alpha+\beta) \tag{4.9}$$

from D_1 and D_2 in a polynomial-size (in $|D_1|$ and $|D_2|$) R(lin)-proof. This concludes the first part of the lemma.

Assume that $\vec{a}$ and $\vec{b}$ consist of coefficients whose absolute values are at most c. Then by a straightforward inspection of the R(lin)-proof of (4.9) from D_1 and D_2 demonstrated above (and by using Lemma 4.2.2 instead of Lemma 4.2.1), one can verify the second part of the lemma. $\square$

An immediate corollary of Lemma 4.2.3 is the efficient formalization in R(lin) (and R^0(lin)) of the following obvious counting argument: If a linear form equals some value in the interval (of integer numbers) $[a_0,a_1]$ and another linear form equals some value in $[b_0,b_1]$ (for some $a_0\le a_1$ and $b_0\le b_1$), then their addition equals some value in $[a_0+b_0,a_1+b_1]$. More formally:

Corollary 4.2.4 *(i) Let z_1 abbreviate $\vec{a}\cdot\vec{x}$ and z_2 abbreviate $\vec{b}\cdot\vec{x}$. Let D_1 be $(z_1=a_0)\vee(z_1=a_0+1)\ldots\vee(z_1=a_1)$, and let D_2 be $(z_2=b_0)\vee(z_2=b_0+1)\ldots\vee(z_2=b_1)$. Then there is a polynomial-size (in the size of D_1, D_2)* R(lin) *proof from D_1, D_2 of*

$$(z_1+z_2=a_0+b_0)\vee(z_1+z_2=a_0+b_0+1)\vee\ldots\vee(z_1+z_2=a_1+b_1)\ . \tag{4.10}$$

(ii) Moreover, assume that the vector $\vec{a}+\vec{b}$ consists of integers with absolute values at most c (which means that D_1, D_2 are $\mathrm{R}_{c,1}$(lin)*-lines), then there is a polynomial-size (in the size of D_1, D_2)* R(lin) *proof of (4.10) from D_1, D_2, with each proof-line being an* $\mathrm{R}_{c,3}$(lin)*-line.*

Lemma 4.2.5

(i) Let $\vec{a}\cdot\vec{x}$ be a linear form with n variables, and let $\mathcal{A} := \{\vec{a}\cdot\vec{x} \mid \vec{x} \in \{0,1\}^n\}$ be the set of all possible values of $\vec{a}\cdot\vec{x}$ over Boolean assignments to $\vec{x}$. Then there is a polynomial-size, in the size of the linear form $\vec{a}\cdot\vec{x}$, R(lin) *proof of*[1]

$$\bigvee_{\alpha\in\mathcal{A}} (\vec{a}\cdot\vec{x} = \alpha)\,. \tag{4.11}$$

(ii) Moreover, if the coefficients in $\vec{a}$ have absolute value at most c, then there is a polynomial-size (in the size of $\vec{a}\cdot\vec{x}$) R(lin) *proof of (4.11) with all proof-lines being* $\mathrm{R}_{c,3}$(lin)*-lines.*

Proof. Without loss of generality, assume that all the coefficients in $\vec{a}$ are nonzero. Consider the Boolean axiom $(x_1 = 0) \vee (x_1 = 1)$ and the (first) coefficient a_1 from $\vec{a}$. Assume that $a_1 \geq 1$. Add $(x_1 = 0)$ to itself a_1 times, and arrive at $(a_1x_1 = 0) \vee (x_1 = 1)$. Then, in the resulted line, add $(x_1 = 1)$ to itself a_1 times, until the following is reached:

$$(a_1x_1 = 0) \vee (a_1x_1 = a_1)\,.$$

Similarly, in case $a_1 \leq -1$ we can subtract ($|a_1| + 1$ many times) $(x_1 = 0)$ from itself in $(x_1 = 0) \vee (x_1 = 1)$, and then subtract ($|a_1| + 1$ many times) $(x_1 = 1)$ from itself in the resulted line.

In the same manner, we can derive the disjunctions: $(a_2x_2 = 0) \vee (a_2x_2 = a_2), \ldots, (a_nx_n = 0) \vee (a_nx_n = a_n)$.

Consider $(a_1x_1 = 0) \vee (a_1x_1 = a_1)$ and $(a_2x_2 = 0) \vee (a_2x_2 = a_2)$. From these two lines, by Lemma 4.2.3, there is a polynomial-size in $|a_1| + |a_2|$ derivation of:

$$(a_1x_1 + a_2x_2 = 0) \vee (a_1x_1 + a_2x_2 = a_1) \vee (a_1x_1 + a_2x_2 = a_2) \vee (a_1x_1 + a_2x_2 = a_1 + a_2)\,. \tag{4.12}$$

In a similar fashion, now consider $(a_3x_3 = 0) \vee (a_3x_3 = a_3)$ and apply again Lemma 4.2.3, to obtain

$$\bigvee_{\alpha\in\mathcal{A}'} (a_1x_1 + a_2x_2 + a_3x_3 = \alpha)\,, \tag{4.13}$$

where $\mathcal{A}'$ are all possible values to $a_1x_1 + a_2x_2 + a_3x_3$ over Boolean assignments to x_1, x_2, x_3. The derivation of (4.13) is of size polynomial in $|a_1| + |a_2| + |a_3|$.

Continue to consider, successively, all other lines $(a_4x_4 = 0) \vee (a_4x_4 = a_4), \ldots, (a_nx_n = 0) \vee (a_nx_n = a_n)$, and apply the same reasoning. Each step uses a derivation of size at most polynomial in $\sum_{i=1}^n |a_i|$. And so overall we reach the desired line (4.11), with a derivation of size polynomial in the size of $\vec{a}\cdot\vec{x}$. This concludes the first part of the lemma.

Assume that $\vec{a}$ consists of coefficients whose absolute value is at most c. Then by a straightforward inspection of the R(lin)-proof demonstrated above (and by using the second part of Lemma 4.2.3), one can see that this proof uses only $\mathrm{R}_{c,3}$(lin)-lines. □

[1]Recall that the size of $\vec{a}\cdot\vec{x}$ is $\sum_{i=1}^n |a_i|$, that is, the size of the unary notation of $\vec{a}$.

Lemma 4.2.6 *For every $n \in \mathbb{N}$, there is a polynomial-size (in n)* $\mathrm{R}^0(\mathrm{lin})$ *proof from*

$$(x_1 = 1) \vee \cdots \vee (x_n = 1) \tag{4.14}$$

of

$$(x_1 + \ldots + x_n = 1) \vee \cdots \vee (x_1 + \ldots + x_n = n)\,. \tag{4.15}$$

Proof. We show that for every $i \in [n]$, there is a polynomial-size (in n) $\mathrm{R}^0(\mathrm{lin})$ proof from $(x_i = 1)$ of $(x_1 + \ldots + x_n = 1) \vee \cdots \vee (x_1 + \ldots + x_n = n)$. This concludes the proof since, by Lemma 4.2.2, we then can derive from (4.14) (with a polynomial-size (in n) $\mathrm{R}^0(\mathrm{lin})$ proof) the disjunction (4.14) in which each $(x_i = 1)$ (for all $i \in [n]$) is replace by $(x_1 + \ldots + x_n = 1) \vee \cdots \vee (x_1 + \ldots + x_n = n)$, which is precisely the disjunction (4.15) (note that (4.15) is an $\mathrm{R}_{1,1}(\mathrm{lin})$-line).

Claim: For every $i \in [n]$, there is a a polynomial-size (in n) $\mathrm{R}^0(\mathrm{lin})$ proof from $(x_i = 1)$ of $(x_1 + \ldots + x_n = 1) \vee \cdots \vee (x_1 + \ldots + x_n = n)$.

Proof of claim: By Lemma 4.2.5, for every $i \in [n]$ there is a polynomial-size (in n) $\mathrm{R}^0(\mathrm{lin})$ proof (using only the Boolean axioms) of

$$(x_1 + \ldots + x_{i-1} + x_{i+1} + \ldots + x_n = 0) \vee \cdots \vee (x_1 + \ldots + x_{i-1} + x_{i+1} + \ldots + x_n = n - 1)\,. \tag{4.16}$$

Now add successively $(x_i = 1)$ to every equation in (4.16) (note that this can be done in $\mathrm{R}^0(\mathrm{lin})$). We obtain precisely $(x_1 + \ldots + x_n = 1) \vee \cdots \vee (x_1 + \ldots + x_n = n)$. $\blacksquare_{\text{Claim}}$ $\square$

Lemma 4.2.7 *There is a polynomial-size (in n)* $\mathrm{R}^0(\mathrm{lin})$ *proof of* $(x_1 + \ldots + x_n = 0) \vee (x_1 + \ldots + x_n = 1)$ *from the collection of disjunctions consisting of* $(x_i = 0) \vee (x_j = 0)$*, for all* $1 \le i < j \le n$.

Proof. We use the less formal reasoning by cases and proceed by induction on n. The base case for $n = 1$ is immediate from the Boolean axiom $(x_1 = 0) \vee (x_1 = 1)$. Assume we already have a polynomial-size proof of

$$(x_1 + \ldots + x_n = 0) \vee (x_1 + \ldots + x_n = 1). \tag{4.17}$$

If $x_{n+1} = 0$ we add $x_{n+1} = 0$ to both of the equations in (4.17), and reach:

$$(x_1 + \ldots + x_{n+1} = 0) \vee (x_1 + \ldots + x_{n+1} = 1). \tag{4.18}$$

Otherwise, $x_{n+1} = 1$, and so we can cut-off $(x_{n+1} = 0)$ in all the initial disjunctions $(x_i = 0) \vee (x_{n+1} = 0)$, for all $1 \le i \le n$. We thus obtain $(x_1 = 0), \ldots, (x_n = 0)$. Adding together $(x_1 = 0), \ldots, (x_n = 0)$ and $(x_{n+1} = 1)$ we arrive at

$$(x_1 + \ldots + x_{n+1} = 1)\,. \tag{4.19}$$

So overall, either (4.18) holds or (4.19) holds; and so (using Lemma 4.2.2) we arrive at the disjunction of (4.19) and (4.18), which is precisely (4.18). $\square$

4.3 Implicational Completeness of R(lin)

In this section we provide a proof of the implicational completeness of R(lin). We shall need this property in the sequel (see Section 4.4.2). Essentially, a system is implicationally complete if whenever something is *semantically* implied by a set of initial premises, then it is also *derivable* from the initial premises. As a consequence, the proof of implicational completeness in this section establishes an alternative completeness proof to that obtained via simulating resolution (see Proposition 4.1.4). Note that we are not concerned in this section with the size of the proofs, but only with their existence.

Formally, we say that R(lin) is *implicationally complete* if for every collection of disjunctions of linear equations $D_0, D_1, \ldots, D_m$, it holds that $D_1, \ldots, D_m \models D_0$ implies that there is an R(lin) proof of D_0 from $D_1, \ldots, D_m$.

Theorem 4.3.1 R(lin) *is implicationally complete.*

Proof. We proceed by induction on n, the number of variables $x_1, \ldots, x_n$ in $D_0, D_1, \ldots, D_m$.

The base case $n = 0$. We need to show that $D_1, \ldots, D_m \models D_0$ implies that there is an R(lin) proof of D_0 from $D_1, \ldots, D_m$, where all D_i's (for $0 \leq i \leq m$) have no variables but only constants. This means that each D_i is a disjunction of equations of the form $(0 = a_0)$ for some integer a_0 (if a linear equation have no variables, then the left hand side of this equation must be 0; see Section 4.1.1).

There are two cases to consider. In the first case D_0 *is satisfiable*. Since D_0 has no variables, this means precisely that D_0 is the equation $(0 = 0)$. Thus, D_0 can be derived easily from any axiom in R(lin) (for instance, by subtracting each equation in $(x_1 = 0) \vee (x_1 = 1)$ from itself, to reach $(0 = 0) \vee (0 = 0)$, which is equal to $(0 = 0)$, since we discard duplicate equations inside disjunctions).

In the second case D_0 *is unsatisfiable*. Thus, since $D_1, \ldots, D_m \models D_0$, there is no assignment satisfying all $D_1, \ldots, D_m$. Hence, there must be at least one unsatisfiable disjunction D_i in $D_1, \ldots, D_m$ (as a disjunction with no variables is either tautological or unsatisfiable). Such an unsatisfiable D_i is a disjunction of zero or more unsatisfiable equations of the form $(0 = a_0)$, for some integer $a_0 \neq 0$. We can then use simplification to cut-off all the unsatisfiable equations in D_i to reach the empty disjunction. By the weakening rule, we can now derive D_0 from the empty disjunction.

The induction step. Assume that the theorem holds for disjunctions with n variables. Let the underlying variables of $D_0, D_1, \ldots, D_m$ be $x_1, \ldots, x_{n+1}$, and assume that

$$D_1, \ldots, D_m \models D_0 \,. \tag{4.20}$$

We write the disjunction D_0 as:

$$\bigvee_{j=1}^{t} \left(\sum_{i=1}^{n} a_i^{(j)} x_i + a_{n+1}^{(j)} x_{n+1} = a_0^{(j)} \right) , \tag{4.21}$$

where the $a_i^{(j)}$'s are integer coefficients. We need to show that there is an R(lin) proof of D_0 from $D_1, \ldots, D_m$.

Let D be a disjunction of linear equations, let x_i be a variable and let $b \in \{0,1\}$. We shall denote by $D\restriction_{x_i=b}$ the disjunction D, where in every equation in D the variable x_i is substituted by b, and the constant terms in the left hand sides of all resulting equations (after substituting b for x_i) switch sides (and change signs, obviously) to the right hand sides of the equations (we have to switch sides of constant terms, as by definition linear equations in R(lin) proofs have all constant terms appearing only on the right hand sides of equations).

We now reason (slightly) informally inside R(lin) (as illustrated in Section 4.2.1). Fix some $b \in \{0,1\}$, and assume that $x_{n+1} = b$. Then, from $D_1, \ldots, D_m$ we can derive (inside R(lin)):

$$D_1\restriction_{x_{n+1}=b}, \ldots, D_m\restriction_{x_{n+1}=b} \ . \tag{4.22}$$

The only variables occurring in (4.22) are $x_1, \ldots, x_n$. From assumption (4.20) we clearly have $D_1\restriction_{x_{n+1}=b}, \ldots, D_m\restriction_{x_{n+1}=b} \models D_0\restriction_{x_{n+1}=b}$. And so by the induction hypothesis there is an R(lin) derivation of $D_0\restriction_{x_{n+1}=b}$ from $D_1\restriction_{x_{n+1}=b}, \ldots, D_m\restriction_{x_{n+1}=b}$. So overall, assuming that $x_{n+1} = b$, there is an R(lin) derivation of $D_0\restriction_{x_{n+1}=b}$ from $D_1, \ldots, D_m$.

We now consider the two possible cases: $x_{n+1} = 0$ and $x_{n+1} = 1$.

In case $x_{n+1} = 0$, by the above discussion, we can derive $D_0\restriction_{x_{n+1}=0}$ from $D_1, \ldots, D_m$. For every $j \in [t]$, add successively ($a_{n+1}^{(j)}$ times) the equation $x_{n+1} = 0$ to the jth equation in $D_0\restriction_{x_{n+1}=0}$ (see (4.21)). We thus obtain precisely D_0.

In case $x_{n+1} = 1$, again, by the above discussion, we can derive $D_0\restriction_{x_{n+1}=1}$ from $D_1, \ldots, D_m$. For every $j \in [t]$, add successively ($a_{n+1}^{(j)}$ times) the equation $x_{n+1} = 1$ to the jth equation in $D_0\restriction_{x_{n+1}=1}$ (recall that we switch sides of constant terms in every linear equation after the substitution of x_{n+1} by 1 is performed in $D_0\restriction_{x_{n+1}=1}$). Again, we obtain precisely D_0. $\square$

4.4 Short Proofs for Hard Tautologies

In this section we show that R^0(lin) is already enough to admit small proofs for "hard" counting principles like the pigeonhole principle and the Tseitin graph formulas for constant degree graphs. On the other hand, as we shall see in Section 4.6, R^0(lin) inherits the same weakness that cutting planes proofs have with respect to the clique-coloring tautologies. Nevertheless, we can efficiently prove the clique-coloring principle in (the stronger system) R(lin), but not by using R(lin) "ability to count", rather by using its (straightforward) ability to simulate Res(2) proofs (that is, resolution proofs extended to operate with 2-DNF formulas, instead of clauses).

4.4.1 The Pigeonhole Principle Tautologies in R^0(lin)

This subsection illustrates polynomial-size R^0(lin) proofs of the pigeonhole principle. This will also allow us to establish polynomial-size multilinear proofs operating with depth-3 multilinear formulas of the pigeonhole principle in the next chapter (Section 5.3).

The *m to n pigeonhole principle* states that m pigeons cannot be mapped one-to-one into $n < m$ holes. As a propositional formula it is usually formulated:

$$\left(\bigwedge_{i\in[m]} \bigvee_{k\in[n]} x_{i,k} \right) \longrightarrow \bigvee_{i<j\in[m]} \bigvee_{k\in[n]} (x_{i,k} \wedge x_{j,k}) \, , \tag{4.23}$$

where each propositional variable $x_{i,j}$ designates that the pigeon i is mapped to the hole j. It is clear that if $m > n$ then PHP^m_n is a tautology.

The negation of (4.23), formulated as an unsatisfiable CNF formula, consists of the following clauses:

$$\begin{array}{ll} \forall i \in [m], & x_{i,1} \vee \ldots \vee x_{i,n} \\ \forall i < j \in [m] \, \forall k \in [n], & \neg x_{i,k} \vee \neg x_{j,k} \end{array} \tag{4.24}$$

This translates (via the translation scheme in Section 4.1.1) into the following unsatisfiable collection of disjunctions of linear equations, denoted $\neg$PHP^m_n:

Definition 4.4.1 *The $\neg$PHP^m_n is the following set of clauses:*

1. *Pigeon axioms:* $(x_{i,1} = 1) \vee \cdots \vee (x_{i,n} = 1)$, *for all* $1 \leq i \leq m$*;*
2. *Hole axioms:* $(x_{i,k} = 0) \vee (x_{j,k} = 0)$, *for all* $1 \leq i < j \leq m$ *and for all* $1 \leq k \leq n$*.*

We now describe a polynomial-size in n refutation of $\neg$PHP^m_n inside R^0(lin). For this purpose it is sufficient to prove a polynomial-size refutation of the pigeonhole principle when the number of pigeons m equals $n + 1$ (because the set of clauses pertaining to $\neg$PHP$^{n+1}_n$ is already contained in the set of clauses pertaining to $\neg$PHP^m_n, for any $m > n$). Thus, we fix $m = n+1$. In this subsection we shall say a proof in R^0(lin) is of *polynomial-size*, always intending *polynomial-size in n* (unless otherwise stated).

By Lemma 4.2.6, for all $i \in [m]$ we can derive from the Pigeon axiom (for the ith pigeon):

$$(x_{i,1} + \ldots + x_{i,n} = 1) \vee \cdots \vee (x_{i,1} + \ldots + x_{i,n} = n) \tag{4.25}$$

with a polynomial-size R^0(lin) proof.

By Lemma 4.2.7, from the Hole axioms we can derive, with a polynomial-size R^0(lin) proof

$$(x_{1,j} + \ldots + x_{m,j} = 0) \vee (x_{1,j} + \ldots + x_{m,j} = 1), \tag{4.26}$$

for all $j \in [n]$.

Let S abbreviate the sum of all formal variables $x_{i,j}$. In other words,

$$S := \sum_{i\in[m],j\in[n]} x_{i,j} \, .$$

Lemma 4.4.2 *There is a polynomial-size* $\mathrm{R}^0(\mathrm{lin})$ *proof from (4.25) (for all $i \in [m]$) of*

$$(S = m) \vee (S = m + 1) \cdots \vee (S = m \cdot n).$$

Proof. For every $i \in [m]$ fix the abbreviation $z_i := x_{i,1} + \ldots + x_{i,n}$. Thus, by (4.25) we have $(z_i = 1) \vee \cdots \vee (z_i = n)$.

Consider $(z_1 = 1) \vee \cdots \vee (z_1 = n)$ and $(z_2 = 1) \vee \cdots \vee (z_2 = n)$. By Corollary 4.2.4, we can derive from these two lines

$$(z_1 + z_2 = 2) \vee (z_1 + z_2 = 3) \vee \cdots \vee (z_1 + z_2 = 2n) \tag{4.27}$$

with a polynomial-size $\mathrm{R}^0(\mathrm{lin})$ proof.

Now, consider $(z_3 = 1) \vee \cdots \vee (z_3 = n)$ and (4.27). By Corollary 4.2.4 again, from these two lines we can derive with a polynomial-size $\mathrm{R}^0(\mathrm{lin})$ proof:

$$(z_1 + z_2 + z_3 = 3) \vee (z_1 + z_2 + z_3 = 4) \vee \cdots \vee (z_1 + z_2 + z_3 = 3n)\,. \tag{4.28}$$

Continuing in the same way, we eventually arrive at

$$(z_1 + \ldots + z_m = m) \vee (z_1 + \ldots + z_m = m + 1) \vee \cdots \vee (z_1 + \ldots + z_m = m \cdot n)\,,$$

which concludes the proof, since S equals $z_1 + \ldots + z_m$. □

Lemma 4.4.3 *There is a polynomial-size* $\mathrm{R}^0(\mathrm{lin})$ *proof from (4.26) of*

$$(S = 0) \vee \cdots \vee (S = n).$$

Proof. For all $j \in [n]$, fix the abbreviation $y_j := x_{1,j} + \ldots + x_{m,j}$. Thus, by (4.26) we have $(y_j = 0) \vee (y_j = 1)$, for all $j \in [n]$. Now the proof is similar to the proof of Lemma 4.2.5, except that here single variables are abbreviations of linear forms.

If $y_1 = 0$ then we can add y_1 to the two sums in $(y_2 = 0) \vee (y_2 = 1)$, and reach $(y_1 + y_2 = 0) \vee (y_1 + y_2 = 1)$ and if $y_1 = 1$ we can do the same and reach $(y_1 + y_2 = 1) \vee (y_1 + y_2 = 2)$. So, by Lemma 4.2.2, we can derive with a polynomial-size $\mathrm{R}^0(\mathrm{lin})$ proof

$$(y_1 + y_2 = 0) \vee (y_1 + y_2 = 1) \vee (y_1 + y_2 = 2)\,. \tag{4.29}$$

Now, we consider the three cases in (4.29): $y_1 + y_2 = 0$ or $y_1 + y_2 = 1$ or $y_1 + y_2 = 2$, and the clause $(y_3 = 0) \vee (y_3 = 1)$. We arrive in a similar manner at $(y_1 + y_2 + y_3 = 0) \vee \cdots \vee (y_1 + y_2 + y_3 = 3)$. We continue in the same way until we arrive at $(S = 0) \vee \cdots \vee (S = n)$. □

Theorem 4.4.4 *There is a polynomial-size* $\mathrm{R}^0(\mathrm{lin})$ *refutation of the m to n pigeonhole principle $\neg PHP^m_n$.*

Proof. By Lemmas 4.4.2 and 4.4.3 above, we need to show a polynomial-size refutation of $(S = m) \vee \cdots \vee (S = m \cdot n)$ and $(S = 0) \vee \cdots \vee (S = n)$.

Since $n < m$, for all $0 \leq k \leq n$, if $S = k$ then using the resolution and simplification rules we can cut-off all the sums in $(S = m) \vee \cdots \vee (S = m \cdot n)$ and arrive at the empty clause. $\square$

4.4.2 Tseitin mod p Tautologies in R^0(lin)

This subsection establishes polynomial-size R^0(lin) proofs of Tseitin graph tautologies (for constant degree graphs). This will also allow us (by the result of the next chapter; see Section 5.3) to extend the multilinear proofs of the Tseitin mod p tautologies shown in Chapter 3 to any field of characteristic 0 (the proofs in Section 3.5 required working over a field containing a primitive pth root of unity when proving the Tseitin mod p tautologies; for more details see Section 5.3).

Recall the Tseitin mod p formulas from the previous chapter (Definition 3.5.1). The following are the clauses of the Tseitin mod p CNF formula (as translated to disjunctions of linear equations). We use the same notation $\mathrm{BTS}_{G,p}$ as in Definition 3.5.1 (though, formally, here we refer to the translation to disjunctions of linear equations, while in Definition 3.5.1 this notation denotes the translation to polynomials).

Definition 4.4.5 (Tseitin mod p formulas ($\mathbf{BTS}_{G,p}$)) *Let $p \geq 2$ be some fixed integer and let $G = (V, E)$ be a connected undirected r-regular graph with n vertices and no multiple edges, and assume that $n \equiv 1 \ (\mathrm{mod}\ p)$. Let $G' = (V, E')$ be the corresponding directed graph that results from G by replacing every (undirected) edge in G with two opposite directed edges.*

Given a vertex $v \in V$, denote the edges in E' coming out of v by $e[v,1], \ldots, e[v,r]$ and define the following set of (translation of) clauses:

$$\mathrm{MOD}_{p,1}(v) := \left\{ \bigvee_{k=1}^{r} (x_{e[v,k],i_k} = 0) \;\middle|\; i_1, \ldots, i_r \in \{0, \ldots, p-1\} \text{ and } \sum_{k=1}^{r} i_k \not\equiv 1 \mod p \right\}.$$

The Tseitin mod p *formula, denoted* $\mathrm{BTS}_{G,p}$*, consists of the following (translation) of* *clauses:*

1. $\bigvee_{i=0}^{p-1}(x_{e,i}=1)$ *, for all* $e\in E'$
(expresses that every edge is assigned at least one value from $0,\dots,p-1$*);*
2. $(x_{e,i}=0)\vee(x_{e,j}=0)$ *, for all* $i\neq j\in\{0,\dots,p-1\}$ *and all* $e\in E'$
(expresses that every edge is assigned at most one value from $0,\dots,p-1$*);*
3. $(x_{e,i}=1)\vee(x_{\bar{e},p-i}=0)$ *and* $(x_{e,i}=0)\vee(x_{\bar{e},p-i}=1)$,[2]
for all two opposite directed edges $e,\bar{e}\in E'$ *and all* $i\in\{0,\dots,p-1\}$
(expresses condition (i) of the Tseitin mod p *principle above);*
4. $\mathrm{MOD}_{p,1}(v)$ *, for all* $v\in V$
(expresses condition (ii) of the Tseitin mod p *principle above).*

Note that for every edge $e\in E'$, the clauses (1,2) in Definition 4.4.5, combined with the Boolean axioms of R^0(lin), force any collection of edge-variables $x_{e,0},\dots,x_{e,p-1}$ to contain exactly one $i\in\{0,\dots,p-1\}$ so that $x_{e,i}=1$. Also, it is easy to verify that, given a vertex $v\in V$, any assignment σ of $0,1$ values (to the relevant variables) satisfies both the disjunctions of (1,2) and the disjunctions of $\mathrm{MOD}_{p,1}(v)$ if and only if σ corresponds to an assignment of values from $\{0,\dots,p-1\}$ to the edges coming out of v that sums up to 1 (mod p).

For the rest of this subsection we fix an integer $p\geq 2$ and a connected undirected r-regular graph $G=(V,E)$ with n vertices and no multiple edges, such that $n\equiv 1 \mod p$ and r is a constant. As in Definition 4.4.5, we let $G'=(V,E')$ be the corresponding directed graph that results from G by replacing every (undirected) edge in G with two opposite directed edges. Note that $\mathrm{BTS}_{G,p}$ consists of only $\mathrm{R}_{1,1}$(lin)-lines (as translations of clauses). We now proceed to refute $\mathrm{BTS}_{G,p}$ inside R^0(lin) with a polynomial-size (in n) refutation.

Given a vertex $v\in V$, and the edges in E' coming out of v, denoted $e[v,1],\dots,e[v,r]$, define the following abbreviation:

$$\alpha_v := \sum_{j=1}^{r}\sum_{i=0}^{p-1} i\cdot x_{e[v,j],i}\,. \tag{4.30}$$

Lemma 4.4.6 *There are constants* c,d *(depending only on* r,p*), such that for any vertex* $v\in V$ *in* G'*, there exists a constant-size (also depending only on* r,p*)* R^0(lin)*-proof from* $\mathrm{BTS}_{G,p}$*, containing only* $\mathrm{R}_{c,d}$(lin)*-lines, of the following disjunction:*

$$\bigvee_{\ell=0}^{r-1}(\alpha_v = 1+\ell\cdot p)\,. \tag{4.31}$$

[2]If $i=0$ then $x_{\bar{e},p-i}$ denotes $x_{\bar{e},0}$.

Proof. Let $T_v \subseteq \mathrm{BTS}_{G,p}$ be the set of all disjunctions of the form (1,2,4) from Definition 4.4.5 that contain only variables pertaining to vertex v (that is, all the variables $x_{e,i}$, where $e \in E'$ is an edge coming out of v, and $i \in \{0, \ldots, p-1\}$).

Claim: T_v semantically implies (4.31), that is:[3]

$$T_v \models \bigvee_{\ell=0}^{r-1} (\alpha_v = 1 + \ell \cdot p) \,.$$

Proof of claim: Let σ be an assignment of $0, 1$ values to the variables in T_v that satisfies both the disjunctions of (1,2) and the disjunctions of $\mathrm{MOD}_{p,1}(v)$ in Definition 4.4.5. As mentioned above (the comment after Definition 4.4.5), such a σ corresponds to an assignment of values from $\{0, \ldots, p-1\}$ to the edges coming out of v that sums up to $1 \mod p$. This means precisely that $\alpha_v = 1 \mod p$ under the assignment σ. Thus, there exists a nonnegative integer k, such that $\alpha_v = 1 + kp$ under σ.

It remains to show that $k \leq r - 1$ (and so the only possible values that α_v can get under σ are $1, 1+p, 1+2p, \ldots, 1+(r-1)p$). Note that because σ gives the value 1 to only one variable from $x_{e[v,j],0}, \ldots, x_{e[v,j],p-1}$ (for every $j \in [r]$), then the maximal value that α_v can have under σ is $r(p-1)$. Thus, $1 + kp \leq rp - r$ and so $k \leq r - 1$. $\blacksquare_{\text{Claim}}$

From Claim 4.4.2 and from the implicational completeness of R(lin) (Theorem 4.3.1), there exists an R(lin) derivation of (4.31) from T_v. This derivation clearly has size depending only on r, p and contains only $\mathrm{R}_{c,d}$(lin)-lines, where c, d are some two integers that depend only on r, p. □

Lemma 4.4.7 *Let $c' = \max\{c, 2p\}$ and $d' = \max\{d, 3\}$, where c, d are the integers taken from Lemma 4.4.6 (note that c', d' depend only on r, p). Then there is a polynomial-size (in n) R^0(lin) derivation from $\mathrm{BTS}_{G,p}$, containing only $\mathrm{R}_{c',d'}$(lin)-lines, of the following disjunction:*

$$\bigvee_{\ell=0}^{(r-1)\cdot n} \left(\sum_{v \in V} \alpha_v = n + \ell \cdot p \right) .$$

Proof. Simply add successively all the equations pertaining to disjunctions (4.31), for all vertices $v \in V$. Formally, we show that for every subset of vertices $\mathcal{V} \subseteq V$, with $|\mathcal{V}| = k$, there is a polynomial-size (in n) R(lin) derivation from $\mathrm{BTS}_{G,p}$ of

$$\bigvee_{\ell=0}^{(r-1)\cdot k} \left(\sum_{v \in \mathcal{V}} \alpha_v = k + \ell \cdot p \right) , \tag{4.32}$$

containing only $\mathrm{R}_{c',d'}$(lin)-lines. Thus, putting $\mathcal{V} = V$, will conclude the proof

[3]Recall that we only consider assignments of $0, 1$ values to variables when considering the semantic implication relation $\models$.

We proceed by induction on the size of $\mathcal{V}$. The base case, $|\mathcal{V}| = 1$, is immediate from Lemma 4.4.6.

Assume that we already derived (4.32) with a polynomial-size (in n) R(lin) proof containing only $\mathrm{R}_{c',d'}$(lin)-lines, for some $\mathcal{V} \subset V$, such that $|\mathcal{V}| = k < n$. Let $u \in V \setminus \mathcal{V}$. By Lemma 4.4.6, we can derive

$$\bigvee_{\ell=0}^{r-1} (\alpha_u = 1 + \ell \cdot p) \tag{4.33}$$

from $\mathrm{BTS}_{G,p}$ with a constant-size proof that contains only $\mathrm{R}_{c,d}$(lin)-lines.

Note that both (4.32) and (4.33) are $\mathrm{R}_{2p,1}$(lin)-lines. Thus, by Lemma 4.2.3, each linear equation in (4.33) can be added to each linear equation in (4.32), with a polynomial-size (in n) R(lin) proof that contains only $\mathrm{R}_{2p,3}$(lin)-lines. This results in the following disjunction:

$$\bigvee_{\ell=0}^{(r-1)\cdot(k+1)} \left(\sum_{v \in \mathcal{V} \cup \{u\}} \alpha_v = k + 1 + \ell \cdot p \right) ,$$

which is precisely what we need to conclude the induction step. □

Given a pair of opposite directed edges $e, \bar{e}$ in G', denote by $T_e \subseteq \mathrm{BTS}_{G,p}$ the set of all disjunctions of the form (1,2,3) from Definition 4.4.5 that contain only variables pertaining to edges $e, \bar{e}$ (that is, all the variables $x_{e,j}, x_{\bar{e},j}$, for all $j \in \{0, \dots, p-1\}$).

Lemma 4.4.8 *There are two integers c'', d'' that depend only on r, p, such that for any pair of opposite directed edges $e, \bar{e}$ in G' and any $i \in \{0, \dots, p-1\}$, there exists a constant-size (depending only on r, p)* R^0(lin) *proof, containing only* $\mathrm{R}_{c'',d''}$(lin)*-lines, from T_e of the following disjunction:*

$$(i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i} = 0) \vee (i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i} = p) \ . \tag{4.34}$$

Proof. First note that T_e semantically implies

$$(x_{e,i} + x_{\bar{e},p-i} = 0) \vee (x_{e,i} + x_{\bar{e},p-i} = 2) \, . \tag{4.35}$$

The number of variables in T_e and (4.35) is constant. Hence, there is a constant-size R^0(lin)-proof of (4.34) from T_e. Also note that

$$\begin{aligned}(x_{e,i} + x_{\bar{e},p-i} = 0) \vee (x_{e,i} + x_{\bar{e},p-i} = 2) \models \\ (i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i} = 0) \vee (i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i} = p) \ .\end{aligned} \tag{4.36}$$

Therefore, there is also an R^0(lin)-proof of constant-size from T_e of the lower line in (4.36) that uses only $\mathrm{R}_{c'',d''}$(lin)-lines, for some two integers c'', d'' depending only on r, p. □

We are now ready to complete the polynomial-size R^0(lin) refutation of $\mathrm{BTS}_{G,p}$. Using the two prior lemmas, the refutation idea is simple, as we now explain. Observe that

$$\sum_{v \in V} \alpha_v = \sum_{\substack{\{e,\bar{e}\} \subseteq E' \\ i \in \{0,\dots,p-1\}}} (i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i}) \ , \tag{4.37}$$

where by $\{e,\bar{e}\} \subseteq E'$ we mean that $e, \bar{e}$ is a pair of opposite directed edges in G'.

Derive by Lemma 4.4.7 the disjunction

$$\bigvee_{\ell=0}^{(r-1)\cdot n} \left(\sum_{v\in V} \alpha_v = n + \ell \cdot p \right) . \tag{4.38}$$

This disjunction expresses the fact that $\sum_{v\in V} \alpha_v = 1 \mod p$ (since $n = 1 \mod p$). On the other hand, using Lemma 4.4.8, we can "sum together" all the equations (4.34) (for all $\{e,\bar{e}\} \subseteq E'$ and all $i \in \{0,\dots,p-1\}$), to obtain a disjunction expressing the statement that

$$\sum_{\substack{\{e,\bar{e}\}\subseteq E' \\ i\in\{0,\dots,p-1\}}} (i \cdot x_{e,i} + (p-i)\cdot x_{\bar{e},p-i}) = 0 \mod p \, .$$

By Equation (4.37), we then obtain the desired contradiction. This idea is formalized in the proof of the following theorem:

Theorem 4.4.9 *Let $G = (V,E)$ be an r-regular graph with n vertices, where r is a constant. Fix some modulus p. Then, there are polynomial-size (in n)* $\mathrm{R}^0(\mathrm{lin})$ *refutations of* $\mathrm{BTS}_{G,p}$.

Proof. Let s,t be two integers (depending only on r,p) defined as $s = \max\{c',c''\}$ and $t = \max\{d',d''\}$, where c',d' are taken from Lemma 4.4.7 and c'',d'' are taken from Lemma 4.4.8.

First, use Lemma 4.4.7 to derive

$$\bigvee_{\ell=0}^{(r-1)\cdot n} \left(\sum_{v\in V} \alpha_v = n + \ell \cdot p \right) , \tag{4.39}$$

using only $\mathrm{R}_{s,t}(\mathrm{lin})$-lines. Second, use Lemma 4.4.8 to derive

$$(i\cdot x_{e,i} + (p-i)\cdot x_{\bar{e},p-i} = p) \vee (i\cdot x_{e,i} + (p-i)\cdot x_{\bar{e},p-i} = 0) \ , \tag{4.40}$$

for every pair of opposite directed edges in $G' = (V,E')$ (as in Definition 4.4.5) and every residue $i \in \{0,\dots,p-1\}$, and using only $\mathrm{R}_{s,t}(\mathrm{lin})$-lines.

We now reason inside $\mathrm{R}^0(\mathrm{lin})$. Pick a pair of opposite directed edges $e,\bar{e}$ and a residue $i \in \{0,\dots,p-1\}$. If $i\cdot x_{e,i} + (p-i)\cdot x_{\bar{e},p-i} = 0$, then subtract this equation successively from every equation in (4.39). We thus obtain a new disjunction, similar to that of (4.39), but which does not contain the $x_{e,i}$ and $x_{\bar{e},p-i}$ variables, and with the same free-terms.

Otherwise, $i\cdot x_{e,i} + (p-i)\cdot x_{\bar{e},p-i} = p$, then subtract this equation successively from every equation in (4.39). Again, we obtain a new disjunction, similar to that of (4.39), but which does not contain the $x_{e,i}$ and $x_{\bar{e},p-i}$ variables, and such that p is subtracted from every free-term in every equation. Since, by assumption, $n \equiv 1 \mod p$, the free-terms in every equation are (still) equal $1 \mod p$.

Now, note that (4.39) is an $\mathrm{R}_{2p,1}$(lin)-line, and (4.40) is an $\mathrm{R}_{p,1}$(lin)-line, and thus the reasoning by (two) cases (that is, $i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i} = 0$ and $i \cdot x_{e,i} + (p-i) \cdot x_{\bar{e},p-i} = p$) as demonstrated above can be done in R^0(lin) using only $\mathrm{R}_{s',t'}$(lin)-lines, where s', t' are two integers depending only on p (these s', t' come from using the reasoning-by-cases which implicitly uses Lemma 4.2.2).

So overall, we applied a polynomial-size R^0(lin)-proofs, containing only $\mathrm{R}_{s',t'}$(lin)-lines, and reached a new $\mathrm{R}_{2p,1}$(lin)-line, in which all the free-terms in the equations equal $1 \mod p$.

We now continue the same process for every pair $e, \bar{e}$ of opposite directed edges in G' and every residue i. Since in each such step we start from an $\mathrm{R}_{2p,1}$(lin)-line and an $\mathrm{R}_{p,1}$(lin)-line (as demonstrated above), then this process can be carried over in R^0(lin) (as demonstrated above, we only use $\mathrm{R}_{s'',t''}$(lin)-lines in this process, for $s'' = \max\{s, s'\}$ and $t'' = \max\{t, t'\}$).

Eventually, we discard all the variables $x_{e,i}$ in the equations, for every $e \in E'$ and $i \in \{0, \ldots, p-1\}$, while all the free-terms in every equation remain to be equal $1 \mod p$. Therefore, we arrive at a disjunction of equations of the form $(0 = \gamma)$ for some $\gamma = 1 \mod p$. By using the simplification rule we can cut-off all such equations, and arrive finally at the empty disjunction. $\square$

4.4.3 The Clique-Coloring Principle in R(lin)

In this section we observe that there are polynomial-size R(lin) proofs of the clique-coloring principle (for certain weak parameters). This implies in particular that R(lin) does not possess the feasible monotone interpolation property (see more details on the interpolation method in Section 4.5).

Atserias, Bonet and Esteban (Atserias et al. (2002)) demonstrated polynomial-size Res(2) refutations of the clique-coloring formulas (for certain weak parameters; see Theorem 4.4.13 below). Thus, it is sufficient to show that R(lin) polynomially-simulates Res(2) proofs (Proposition 4.4.12). This can be shown in a straightforward manner. As noted in the first paragraph of Section 4.4, because the proofs of the clique-coloring formula we discuss here only follow the proofs inside Res(2), then in fact these proofs do not take any advantage of the capability to "count" inside R(lin) (this capability is exemplified, for instance, in Section 4.2.2).

We start with the clique-coloring formulas (these formulas will also be used in Section 4.6). These formulas express the clique-coloring principle that has been widely used in the proof complexity literature (cf., Bonet et al. (1997), Pudlák (1997), Krajíček (1997), Krajíček (1998), Atserias et al. (2002), Krajíček (2007)). This principle is based on the following basic combinatorial idea. Let $G = (V, E)$ be an undirected graph with n vertices and let $k' < k$ be two integers. Then, one of the following must hold:

(i) The graph G does not contain a *clique with k vertices*;

(ii) The graph G is not a *complete k'-partite graph.* In other words, there is no way to partition G into k' subgraphs $G_1, \dots, G_{k'}$, such that every G_i is an independent set, and for all $i \neq j \in [k']$, all the vertices in G_i are connected by edges (in E) to all the vertices in G_j.

Obviously, if Item (ii) above is false (that is, if G is a complete k'-partite graph), then there exists a k'-coloring of the vertices of G; hence the name *clique-coloring* for the principle.

The propositional formulation of the (negation of the) clique-coloring principle is as follows. Each variable $p_{i,j}$, for all $i \neq j \in [n]$, is an indicator variable for the fact that there is an edge in G between vertex i and vertex j. Each variable $q_{\ell,i}$, for all $\ell \in [k]$ and all $i \in [n]$, is an indicator variable for the fact that the vertex i in G is the ℓth vertex in the k-clique. Each variable $r_{\ell,i}$, for all $\ell \in [k']$ and all $i \in [n]$, is an indicator variable for the fact that the vertex i in G is in the independent set G_ℓ.

Definition 4.4.10 (The Clique-Coloring formulas) *The negation of the clique-coloring principle consists of the following unsatisfiable collection of clauses (as translated to disjunctions of linear equations), denoted* $\neg\text{CLIQUE}^n_{k,k'}$*:*

i. $(q_{\ell,1} = 1) \vee \cdots \vee (q_{\ell,n} = 1)$, *for all* $\ell \in [k]$

(expresses that there exists at least one vertex in G which constitutes the ℓth vertex of the k-clique);

ii. $(q_{\ell,i} = 0) \vee (q_{\ell,j} = 0)$, *for all* $i \neq j \in [n],\ \ell \in [k]$

(expresses that there exists at most one vertex in G which constitutes the ℓth vertex of the k-clique);

iii. $(q_{\ell,i} = 0) \vee (q_{\ell',i} = 0)$, *for all* $i \in [n],\ \ell \neq \ell' \in [k]$

(expresses that the ith vertex of G cannot be both the ℓth and the ℓ'th vertex of the k-clique);

iv. $(q_{\ell,i} = 0) \vee (q_{\ell',j} = 0) \vee (p_{i,j} = 1)$, *for all* $\ell \neq \ell' \in [k], i \neq j \in [n]$

(expresses that if both the vertices i and j in G are in the k-clique, then there is an edge in G between i and j);

v. $(r_{1,i} = 1) \vee \cdots \vee (r_{k',i} = 1)$, *for all* $i \in [n]$

(expresses that every vertex of G is in at least one independent set);

vi. $(r_{\ell,i} = 0) \vee (r_{\ell',i} = 0)$, *for all* $\ell \neq \ell \in [k'], i \in [n]$

(expresses that every vertex of G pertains to at most one independent set);

vii. $(p_{i,j} = 0) \vee (r_{t,i} = 0) \vee (r_{t,j} = 0)$, *for all* $i \neq j \in [n], t \in [k']$

(expresses that if there is an edge between vertex i and j in G, then i and j cannot be in the same independent set);

Remark: Our formulation of the clique-coloring formulas above is similar to the one used by Bonet et al. (1997), except that we consider also the $p_{i,j}$ variables (we added the (iv) clauses and changed accordingly the (vii) clauses). This is done for the sake of clarity of the contradiction itself, and also to make it clear that the formulas are in the appropriate form required by the interpolation method (see Section 4.5 for details on the interpolation method). By resolving over the $p_{i,j}$ variables in (iv) and (vii), one can obtain precisely the collection of clauses in Bonet et al. (1997).

Atserias, Bonet and Esteban (Atserias et al. (2002)) demonstrated polynomial-size (in n) Res(2) refutations of $\neg\text{CLIQUE}^n_{k,k'}$, when $k = \sqrt{n}$ and $k' = (\log n)^2/8 \log\log n$. These are rather weak parameters, but they suffice to establish the fact that Res(2) does not possess the feasible monotone interpolation property.

The Res(2) proof system (also called 2-*DNF resolution*), first considered in Krajíček (2001), is resolution extended to operate with 2-DNF formulas, defined as follows.

A 2-*term* is a conjunction of up to two literals. A 2-DNF is a disjunction of 2-terms. The size of a 2-term is the number of literals in it (that is, either 1 or 2). The *size of a* 2-*DNF* is the total size of all the 2-terms in it.

Definition 4.4.11 (Res(2)) *A* Res(2) proof of a 2-DNF D from a collection K of 2-DNFs *is a sequence of* 2*-DNFs* $D_1, D_2, \dots, D_s$*, such that* $D_s = D$*, and every* D_j *is either from* K *or was derived from previous line(s) in the sequence by the following inference rules:*

Cut *Let* A, B *be two* 2*-DNFs.*

From $A \vee \bigwedge_{i=1}^{2} l_i$ *and* $B \vee \bigvee_{i=1}^{2} \neg l_i$ *derive* $A \vee B$*, where the* l_i*'s are (not necessarily distinct) literals (and* $\neg l_i$ *is the negation of the literal* l_i*).*

AND-introduction *Let* A, B *be two* 2*-DNFs and* l_1, l_2 *two literals.*

From $A \vee l_1$ *and* $B \vee l_2$ *derive* $A \vee B \vee \bigwedge_{i=1}^{2} l_i$*.*

Weakening *From a* 2*-DNF* A *derive* $A \vee \bigwedge_{i=1}^{2} l_i$*, where the* l_i*'s are (not necessarily distinct) literals.*

A Res(2) refutation *of a collection of* 2*-DNFs* K *is a Res*(2) *proof of the empty disjunction* $\Box$ *from* K *(the empty disjunction stands for* FALSE*). The* size *of a Res*(2) *proof is the total size of all the* 2*-DNFs in it.*

Given a collection K of 2-DNFs we translate it into a collection of disjunctions of linear equations via the following translation scheme. For a literal l, denote by $\widehat{l}$ the translation that maps a variable x_i into x_i, and $\neg x_i$ into $1 - x_i$. A 2-term $l_1 \wedge l_2$ is first transformed into the equation $\widehat{l_1} + \widehat{l_2} = 2$, and then changing sides of the free-terms so that the final translation of $l_1 \wedge l_2$ has only a single free-term in the right hand side. A disjunction of 2-terms (that is, a 2-DNF) $D = \bigvee_{i \in I}(l_{i,1} \wedge l_{i,2})$ is translated into the disjunction of the translations of the 2-terms, denoted by $\widehat{D}$. It is clear that every assignment satisfies a 2-DNF D if and only if it satisfies $\widehat{D}$.

Proposition 4.4.12 R(lin) *polynomially simulates Res(2). In other words, if π is a Res(2) proof of D from a collection of 2-DNFs $K_1, \ldots, K_t$, then there is an* R(lin) *proof of $\widehat{D}$ from $\widehat{K}_1, \ldots, \widehat{K}_t$ whose size is polynomial in the size of π.*

The proof of Proposition 4.4.12 proceeds by induction on the length (that is, the number of proof-lines) in the Res(2) proof. This is pretty straightforward and similar to the simulation of resolution by R(lin), as illustrated in the proof of Proposition 4.1.4. We omit the details.

Theorem 4.4.13 (Atserias et al. (2002)) *Let $k = \sqrt{n}$ and $k' = (\log n)^2/8 \log\log n$. Then $\neg\text{CLIQUE}^n_{k,k'}$ has Res(2) refutations of size polynomial in n.*

Thus, Proposition 4.4.12 yields the following:

Corollary 4.4.14 *Let k, k' be as in Theorem 4.4.13. Then $\neg\text{CLIQUE}^n_{k,k'}$ has* R(lin) *refutations of size polynomial in n.*

The following corollary is important (we refer the reader to Section 4.5.3 for the necessary relevant definitions concerning the *feasible monotone interpolation property*).

Corollary 4.4.15 R(lin) *does not possess the feasible monotone interpolation property.*

Remark: The proof of $\neg\text{CLIQUE}^n_{k,k'}$ inside Res(2) demonstrated in Atserias et al. (2002) (and hence, also the corresponding proof inside R(lin)) proceeds along the following lines. First reduce $\neg\text{CLIQUE}^n_{k,k'}$ to the k to k' pigeonhole principle. For the appropriate values of the parameters k and k' — and specifically, for the values in Theorem 4.4.13 — there is a short *resolution* proof of the k to k' pigeonhole principle (this was shown in Buss and Pitassi (1997)); (this resolution proof is polynomial in the number of pigeons k, but not in the number of holes k', which is exponentially smaller than k).[4] Therefore, in order to conclude the refutation of $\neg\text{CLIQUE}^n_{k,k'}$ inside Res(2) (or inside R(lin)), it suffices to simulate the short resolution refutation of the k to k' pigeonhole principle. **It is important to emphasize this point:** After reducing, inside R(lin), $\neg\text{CLIQUE}^n_{k,k'}$ to the pigeonhole principle, one simulates the *resolution* refutation of the pigeonhole principle, and this has nothing to do with the small-size R^0(lin) refutations of the pigeonhole principle demonstrated in Section 4.4.1. This is because, the reduction (inside R(lin)) of $\neg\text{CLIQUE}^n_{k,k'}$ to the k to k' pigeonhole principle, results in a *substitution instance* of the pigeonhole principle formulas; in other words, the reduction results in a collection of disjunctions that are similar to the pigeonhole principle disjunctions *where each original pigeonhole principle variable is substituted by some big formula* (and, in particular, there are no two integers c, d independent from the number of variables, such that these disjunctions are all R$_{c,d}$(lin)-lines). (Note that R^0(lin) does not admit short proofs of the clique-coloring formulas as we show in Section 4.6.)

[4]Whenever $k \geq 2k'$ the k to k' pigeonhole principle is referred to as the *weak pigeonhole principle*.

4.5 Interpolation Results for R^0(lin)

In this section we study the applicability of the feasible (non-monotone) interpolation technique to R^0(lin) refutations. In particular, we show that R^0(lin) admits a polynomial (in terms of the R^0(lin)-proofs) upper bound on the (non-monotone) circuit-size of interpolants. In the next section we shall give a polynomial upper bound on the *monotone* circuit-size of interpolants, but only in the case that the interpolant corresponds to the clique-coloring formulas (whereas, in this section we are interested in the general case; that is, upper bounding circuit-size of interpolants corresponding to any formula [of the prescribed type; see below]). First, we shortly describe the feasible interpolation method and explain how this method can be applied to obtain (sometime, conditional) lower bounds on proof size. Explicit usage of the interpolation method in proof complexity goes back to Krajíček (1994).

Let $A_i(\vec{p},\vec{q})$, $i \in I$, and $B_j(\vec{p},\vec{r})$, $j \in J$, (I and J are sets of indices) be a collection of formulas (for instance, a collection of disjunctions of linear equations) in the displayed variables only. Denote by $A(\vec{p},\vec{q})$ the conjunction of all $A_i(\vec{p},\vec{q})$, $i \in I$, and by $B(\vec{p},\vec{r})$, the conjunction of all $B_j(\vec{p},\vec{r})$, $j \in J$. Assume that $\vec{p},\vec{q},\vec{r}$ are pairwise disjoint sets of distinct variables, and that there is no assignment that satisfies both $A(\vec{p},\vec{q})$ and $B(\vec{p},\vec{r})$. Fix an assignment $\vec{\alpha}$ to the variables in $\vec{p}$. The $\vec{p}$ variables are the *only common variables* of the A_i's and the B_j's. Therefore, either $A(\vec{\alpha},\vec{q})$ is unsatisfiable or $B(\vec{\alpha},\vec{r})$ is unsatisfiable.

The interpolation technique transforms a refutation of $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$, in some proof system, into a circuit (usually a Boolean circuit) that outputs 1 for those assignments $\vec{\alpha}$ (for $\vec{p}$) for which $A(\vec{\alpha},\vec{q})$ is unsatisfiable, and outputs 0 for those assignments $\vec{\alpha}$ for which $B(\vec{\alpha},\vec{r})$ is unsatisfiable (the two cases are not necessarily exclusive, so if both cases hold for an assignment, the circuit can output either that the first case holds or that the second case holds). In other words, given a refutation of $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$, we construct a circuit $C(\vec{p})$, called *the interpolant*, such that

$$\begin{array}{lcll} C(\vec{\alpha}) = 1 & \Longrightarrow & A(\vec{\alpha},\vec{q}) & \text{is unsatisfiable, and} \\ C(\vec{\alpha}) = 0 & \Longrightarrow & B(\vec{\alpha},\vec{r}) & \text{is unsatisfiable.} \end{array} \tag{4.41}$$

(Note that if U denotes the set of those assignments $\vec{\alpha}$ for which $A(\vec{\alpha},\vec{q})$ is *satisfiable*, and V denotes the set of those assignments $\vec{\alpha}$ for which $B(\vec{\alpha},\vec{r})$ is *satisfiable*, then U and V are disjoint [since $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ is unsatisfiable], and $C(\vec{p})$ separates U from V; see Definition 4.5.2 below.)

Assume that for a proof system $\mathcal{P}$ the transformation from refutations of $A(\vec{p},\vec{q}), B(\vec{p},\vec{r})$ into the corresponding interpolant circuit $C(\vec{p})$ results in a circuit whose size is polynomial in the size of the refutation. Then, an exponential lower bound on circuits for which (4.41) holds, implies an exponential lower bound on $\mathcal{P}$-refutations of $A(\vec{p},\vec{q}), B(\vec{p},\vec{r})$.

4.5.1 Interpolation for Semantic Refutations

We now lay out the basic concepts needed to formally describe the feasible interpolation technique. We use the general notion of *semantic refutations* (which generalizes any standard propositional refutation system). We shall use a close terminology to that in Krajíček (1997).

Definition 4.5.1 (Semantic refutation) *Let N be a fixed natural number and let $E_1, \ldots, E_k \subseteq \{0,1\}^N$, where $\bigcap_{i=1}^{k} E_i = \emptyset$. A* semantic refutation *from $E_1, \ldots, E_k$ is a sequence $D_1, \ldots, D_m \subseteq \{0,1\}^N$ with $D_m = \emptyset$ and such that for every $i \in [m]$, D_i is either one of the E_j's or is deduced from two previous D_j, D_ℓ, $1 \leq j, \ell < i$, by the following* semantic inference rule*:*

- *From $A, B \subseteq \{0,1\}^N$ deduce any C, such that $C \supseteq (A \cap B)$.*

Observe that any standard propositional refutation (with inference rules that derive from at most two proof-lines, a third line) can be regarded as a semantic refutation: just substitute each refutation-line by the set of its satisfying assignments; and by the soundness of the inference rules applied in the refutation, it is clear that each refutation-line (considered as the set of assignments that satisfy it) is deduced by the semantic inference rule from previous refutation-lines.

Definition 4.5.2 (Separating circuit) *Let $\mathcal{U}, \mathcal{V} \subseteq \{0,1\}^n$, where $\mathcal{U} \cap \mathcal{V} = \emptyset$, be two disjoint sets. A Boolean circuit C with n input variables is said to* separate $\mathcal{U}$ from $\mathcal{V}$ *if $C(\vec{x}) = 1$ for every $\vec{x} \in \mathcal{U}$, and $C(\vec{x}) = 0$ for every $\vec{x} \in \mathcal{V}$. In this case we also say that $\mathcal{U}$ and $\mathcal{V}$ are* separated by C.

Convention: In what follows we sometime identify a Boolean formula with the set of its satisfying assignments.

Notation: For two (or more) binary strings $u, v \in \{0,1\}^*$, we write (u, v) to denote the concatenation of the u with v.

Let $N = n + s + t$ be fixed from now on. Let $A_1, \ldots, A_k \subseteq \{0,1\}^{n+s}$ and let $B_1, \ldots, B_\ell \subseteq \{0,1\}^{n+t}$. Define the following two sets of assignments of length n (formally, $0, 1$ strings of length n) that can be extended to satisfying assignments of $A_1, \ldots, A_k$ and $B_1, \ldots, B_\ell$, respectively (formally, those $0, 1$ string of length $n + s$ and $n + t$ that are contained in all $A_1, \ldots, A_k$ and $B_1, \ldots, B_\ell$, respectively):

$$\mathcal{U}_A := \left\{ u \in \{0,1\}^n \;\middle|\; \exists q \in \{0,1\}^s,\, (u, q) \in \bigcap_{i=1}^{k} A_i \right\},$$

$$\mathcal{V}_B := \left\{ v \in \{0,1\}^n \;\middle|\; \exists r \in \{0,1\}^t,\, (v, r) \in \bigcap_{i=1}^{\ell} B_i \right\}.$$

Definition 4.5.3 (Polynomial upper bounds on interpolants) *Let $\mathcal{P}$ be a propositional refutation system. Assume that $\vec{p}, \vec{q}, \vec{r}$ are pairwise disjoint sets of distinct variables, where $\vec{p}$ has n variables, $\vec{q}$ has s variables and $\vec{r}$ has t variables. Let $A_1(\vec{p}, \vec{q}), \ldots, A_k(\vec{p}, \vec{q})$ and $B_1(\vec{p}, \vec{r}), \ldots, B_\ell(\vec{p}, \vec{r})$ be two collections of formulas with the displayed variables only. Assume that for any such $A_1(\vec{p}, \vec{q}), \ldots, A_k(\vec{p}, \vec{q})$ and $B_1(\vec{p}, \vec{r}), \ldots, B_\ell(\vec{p}, \vec{r})$, if there exists a $\mathcal{P}$-refutation of size S for $A_1(\vec{p}, \vec{q}) \wedge \cdots \wedge A_k(\vec{p}, \vec{q}) \wedge B_1(\vec{p}, \vec{r}) \wedge \ldots \wedge B_\ell(\vec{p}, \vec{r})$ then there exists a Boolean circuit separating $\mathcal{U}_A$ from $\mathcal{V}_B$ of size polynomial in S.*[5] *In this case we say that $\mathcal{P}$ has a* polynomial upper bound on interpolant circuits.

4.5.1.1 The Communication Game Technique

The *feasible interpolation via communication game technique* is based on transforming proofs into Boolean circuits, where the size of the resulting circuit depends on the communication complexity of each proof-line. This technique goes back to Impagliazzo et al. (1994) and Razborov (1995) and was subsequently applied and extended in Bonet et al. (1997) and Krajíček (1997) (Impagliazzo et al. (1994) and Bonet et al. (1997) did not use explicitly the notion of interpolation of tautologies or contradictions). We shall employ the interpolation theorem of Krajíček in Krajíček (1997) that demonstrates how to transform a small semantic refutation with each proof-line having low communication complexity into a small Boolean circuit separating the corresponding sets.

The underlying idea of the interpolation via communication game technique is that a (semantic) refutation, where each proof-line is of small (that is, logarithmic) communication complexity, can be transformed into an efficient communication protocol for the *Karchmer-Wigderson game* (following Karchmer and Wigderson (1988)) for two players. In the Karchmer-Wigderson game the first player knows some binary string $u \in U$ and the second player knows some different binary string $v \in V$, where U and V are disjoint sets of strings. The two players communicate by sending information bits to one another (following a protocol previously agreed on). The goal of the game is for the two players to decide on an index i such that the ith bit of u is different from the ith bit of v. An efficient Karchmer-Wigderson protocol (by which we mean a protocol that requires the players to exchange at most a logarithmic number of bits in the worst-case) can then be transformed into a small circuit separating U from V (see Definition 4.5.2). This efficient transformation from protocols for Karchmer-Wigderson games (described in a certain way) into circuits, was demonstrated by Razborov in Razborov (1995). So overall, given a semantic refutation with proof-lines of low communication complexity, one can obtain a small circuit for separating the corresponding sets.

First, we need to define the concept of *communication complexity* in a suitable way for the interpolation theorem.

Definition 4.5.4 (Communication complexity; Krajíček (1997), Definition 4.3) *Let $N = n + s + t$ and $A \subseteq \{0, 1\}^N$. Let $u, v \in \{0, 1\}^n$, $q^u \in \{0, 1\}^s$, $r^v \in \{0, 1\}^t$. Denote*

[5]Here $\mathcal{U}_A$ and $\mathcal{V}_B$ are defined as above, by identifying the $A_i(\vec{p}, \vec{q})$'s and the $B_i(\vec{p}, \vec{r})$'s with the sets of assignments that satisfy them.

by u_i, v_i the ith bit of u, v, respectively, and let (u, q^u, r^v) and (v, q^u, r^v) denote the concatenation of strings u, q^u, r^v and v, q^u, r^v, respectively. Consider the following three tasks:

1. *Decide whether $(u, q^u, r^v) \in A$;*
2. *Decide whether $(v, q^u, r^v) \in A$;*
3. *If one of the following holds:*
   (i) $(u, q^u, r^v) \in A$ *and* $(v, q^u, r^v) \notin A$*; or*
   (ii) $(u, q^u, r^v) \notin A$ *and* $(v, q^u, r^v) \in A$*,*
   then find an $i \in [n]$, such that $u_i \neq v_i$;

Consider a game between two players, Player I and Player II, where Player I knows $u \in \{0,1\}^n, q^u \in \{0,1\}^s$ and Player II knows $v \in \{0,1\}^n, r^v \in \{0,1\}^t$. The two players communicate by exchanging bits of information between them (following a protocol previously agreed on). The communication complexity of A, *denoted $CC(A)$, is the minimal (over all protocols) number of bits that players I and II need to exchange in the worst-case in solving each of Tasks 1, 2 and 3 above.*[6]

For $A \subseteq \{0,1\}^{n+s}$ define

$$\dot{A} := \left\{(a,b,c) \mid (a,b) \in A \text{ and } c \in \{0,1\}^t\right\} ,$$

where a and b range over $\{0,1\}^n$ and $\{0,1\}^s$, respectively. Similarly, for $B \subseteq \{0,1\}^{n+t}$ define

$$\dot{B} := \left\{(a,b,c) \mid (a,c) \in B \text{ and } b \in \{0,1\}^t\right\} ,$$

where a and c range over $\{0,1\}^n$ and $\{0,1\}^t$, respectively.

Theorem 4.5.5 (Krajíček (1997), Theorem 5.1) *Let $A_1, \ldots, A_k \subseteq \{0,1\}^{n+s}$ and $B_1, \ldots, B_\ell \subseteq \{0,1\}^{n+t}$. Let $D_1, \ldots, D_m$ be a semantic refutation from $\dot{A}_1, \ldots, \dot{A}_k$ and $\dot{B}_1, \ldots, \dot{B}_\ell$. Assume that $CC(D_i) \leq \zeta$, for all $i \in [m]$. Then, the sets $\mathcal{U}_A$ and $\mathcal{V}_B$ (as defined above) can be separated by a Boolean circuit of size $(m+n)2^{O(\zeta)}$.*

4.5.2 Polynomial Upper Bounds on Interpolants for R^0(lin)

Here we apply Theorem 4.5.5 to show that R^0(lin) has polynomial upper bounds on its interpolant circuits. Again, in what follows we sometime identify a disjunction of linear equations with the set of its satisfying assignments.

[6]In other words, $CC(A)$ is the minimal number ζ, for which there exists a protocol, such that for every input (u, q^u to Player I and v, r^v to Player II) and every task (from Tasks 1, 2 and 3), the players need to exchange at most ζ bits in order to solve the task.

Theorem 4.5.6 $\mathrm{R}^0(\mathrm{lin})$ *has a polynomial upper bound on interpolant circuits (Definition 4.5.3).*

In light of Theorem 4.5.5, to demonstrate that a certain propositional refutation system $\mathcal{P}$ possesses a polynomial upper bound on interpolant circuits (see Definition 4.5.3) it suffices to show that any proof-line of $\mathcal{P}$ induces a set of assignments with at most a logarithmic (in the number of variables) communication complexity (Definition 4.5.4). Therefore, all we need in order to establish Theorem 4.5.6 is the following lemma:

Lemma 4.5.7 *Let k, d be two constants, let D be an $\mathrm{R}_{k,d}(\mathrm{lin})$-line with N variables, and let $\widetilde{D}$ be the set of assignments that satisfy D. Then, $CC(\widetilde{D}) = O(\log N)$.*

Proof. Let $N = n + s + t$ (and so $\widetilde{D} \in \{0,1\}^{n+s+t}$). For the sake of convenience we shall assume that the N variables in D are partitioned into (pairwise disjoint) three groups $\vec{p} := (p_1 \ldots, p_n)$, $\vec{q} := (q_1, \ldots, q_s)$ and $\vec{r} := (r_1, \ldots, r_t)$. Let $u, v \in \{0,1\}^n$, $q^u \in \{0,1\}^s$, $r^v \in \{0,1\}^t$. Assume that Player I knows u, q^u and Player II knows v, r^v.

By assumption, we can partition the disjunction D into a *constant number* d of disjuncts, where one disjunct is a (possibly empty, translation of a) clause in the $\vec{p}, \vec{q}, \vec{r}$ variables (see Section 4.1.1), and all other disjuncts have the following form:

$$\bigvee_{i \in I} \left(\vec{a} \cdot \vec{p} + \vec{b} \cdot \vec{q} + \vec{c} \cdot \vec{r} = \ell_i \right) , \tag{4.42}$$

where I is (an unbounded) set of indices, ℓ_i are integer numbers, for all $i \in I$, and $\vec{a}, \vec{b}, \vec{c}$ denote vectors of n, s and t constant coefficients (that is, all the coefficients have an absolute value at most k), respectively.

Let us denote the (translation of the) clause from D in the $\vec{p}, \vec{q}, \vec{r}$ variables by

$$P \vee Q \vee R \, ,$$

where P, Q and R denote the (translated) sub-clauses consisting of the $\vec{p}$, $\vec{q}$ and $\vec{r}$ variables, respectively.

We need to show that by exchanging $O(\log N)$ bits, the players can solve each of Tasks 1, 2 and 3 from Definition 4.5.4, correctly.

Task 1: The players need to decide whether $(u, q^u, r^v) \in \widetilde{D}$. Player II, who knows r^v, computes the numbers $\vec{c} \cdot r^v$, for every $\vec{c}$ pertaining to every disjunct of the form shown in Equation (4.42) above. Then, Player II sends the (binary representation of) these numbers to Player I. Since there are only a constantly many such numbers and the coefficients in every $\vec{c}$ are also constants, this amounts to $O(\log t) \leq O(\log N)$ bits that Player II sends to Player I. Player II also computes the truth value of the sub-clause R, and sends this (single-bit) value to Player I.

Now, it is easy to see that Player I has sufficient data to compute by herself/himself whether $(u, q^u, r^v) \in \widetilde{D}$ (Player I can then send a single bit informing Player II whether $(u, q^u, r^v) \in \widetilde{D}$).

Task 2: This is analogous to Task 1.

Task 3: Assume that $(u, q^u, r^v) \in \widetilde{D}$ and $(v, q^u, r^v) \notin \widetilde{D}$ (the case $(u, q^u, r^v) \notin \widetilde{D}$ and $(v, q^u, r^v) \in \widetilde{D}$ is analogous).

The first rounds of the protocol are completely similar to that described in Task 1 above: Player II, who knows r^v, computes the numbers $\vec{c} \cdot r^v$, for every $\vec{c}$ pertaining to every disjunct of the form shown in Equation (4.42) above. Then, Player II sends the (binary representation of) these numbers to Player I. Player II also computes the truth value of the sub-clause R, and sends this (single-bit) value to Player I. Again, this amounts to $O(\log N)$ bits that Player II sends to Player I.

By assumption (that $(u, q^u, r^v) \in \widetilde{D}$ and $(v, q^u, r^v) \notin \widetilde{D}$) the players need to deal only with the following two cases:

Case 1: The assignment (u, q^u, r^v) satisfies the clause $P \vee Q \vee R$ while (v, q^u, r^v) falsifies $P \vee Q \vee R$. Thus, it must be that $\vec{u}$ satisfies the sub-clause P while $\vec{v}$ falsifies P. This means that for any $i \in [n]$ such that u_i sets to 1 a literal in P (there ought to exist at least one such i), it must be that $u_i \neq v_i$. Therefore, all that Player I needs to do is to send the (binary representation of) index i to Player II. (This amounts to $O(\log N)$ bits that Player I sends to Player II.)

Case 2: There is some linear equation

$$\vec{a} \cdot \vec{p} + \vec{b} \cdot \vec{q} + \vec{c} \cdot \vec{r} = \ell \tag{4.43}$$

in D, such that $\vec{a} \cdot u + \vec{b} \cdot q^u + \vec{c} \cdot r^v = \ell$. Note that (by assumption that $(v, q^u, r^v) \notin \widetilde{D}$) it must also hold that: $\vec{a} \cdot v + \vec{b} \cdot q^u + \vec{c} \cdot r^v \neq \ell$ (and so there is an $i \in [n]$, such that $u_i \neq v_i$). Player I can find linear equation (4.43), as he/she already received from Player II all the possible values of $\vec{c} \cdot \vec{r}$ (for all possible $\vec{c}$'s in D).

Recall that the left hand side of a linear equation $\vec{d} \cdot \vec{x} = \ell$ is called the *linear form* of the equation. By assumption, there are only constant d many distinct linear forms in D. Since both players know these linear forms, we can assume that each linear form has some index associated to it by both players. Player I sends to Player II the index of the linear form $\vec{a} \cdot \vec{p} + \vec{b} \cdot \vec{q} + \vec{c} \cdot \vec{r}$ from (4.43) in D. Since there are only constantly many such linear forms in D, it takes only constant number of bits to send this index.

Now both players need to apply a protocol for finding an $i \in [n]$ such that $u_i \neq v_i$, where $\vec{a} \cdot \vec{u} + \vec{b} \cdot q^u + \vec{c} \cdot r^v = \ell$ and $\vec{a} \cdot \vec{v} + \vec{b} \cdot q^u + \vec{c} \cdot r^v \neq \ell$. Thus, it remains only to prove the following claim:

Claim: There is a communication protocol in which Player I and Player II need at most $O(\log N)$ bits of communication in order to find an $i \in [n]$ such that $u_i \neq v_i$ (under the above conditions).

Proof of claim: We invoke the well-known connection between Boolean circuit-depth and communication complexity. Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function. Denote by $\mathrm{dp}(f)$ the minimal depth of a bounded fan-in Boolean circuit computing f. Consider a game between two players: Player I knows some $\vec{x} \in \{0,1\}^N$ and Player II knows some other $\vec{y} \in \{0,1\}^N$, such that $f(\vec{x}) = 1$ while $f(\vec{y}) = 0$. The goal of the game is to find an $i \in [N]$ such that $x_i \neq y_i$. Denote by $\mathrm{CC}'(f)$ the minimal number of bits needed for the two players to communicate (in the worst case[7]) in order to solve this game.[8] Then, for any function f it is known that $\mathrm{dp}(f) = \mathrm{CC}'(f)$ (see Karchmer and Wigderson (1988)).

Therefore, to conclude the proof of the claim it is enough to establish that the function $f : \{0,1\}^N \to \{0,1\}$ that receives the input variables $\vec{p}, \vec{q}, \vec{r}$ and computes the truth value of $\vec{a} \cdot \vec{p} + \vec{b} \cdot \vec{q} + \vec{c} \cdot \vec{r} = \ell$ has Boolean circuit of depth $O(\log N)$. In case all the coefficients in $\vec{a}, \vec{b}, \vec{c}$ are 1, it is easy to show[9] that there is a Boolean circuit of depth $O(\log N)$ that computes the function f. In the case that the coefficients in $\vec{a}, \vec{b}, \vec{c}$ are all constants, it is easy to show, by a reduction to the case where all coefficients are 1, that there is a Boolean circuit of depth $O(\log N)$ that computes the function f. We omit the details. $\blacksquare_{\text{Claim}}$ $\square$

4.5.3 Feasible Monotone Interpolation

Here we formally define the feasible monotone interpolation property to complement Theorem 4.4.15. The definition is taken mainly from Krajíček (1997). Recall that for two binary strings of length n (or equivalently, Boolean assignments for n propositional variables), α, α', we denote by $\alpha' \geq \alpha$ that α' is *bitwise* greater than α, that is, that for all $i \in [n]$, $\alpha'_i \geq \alpha_i$ (where α'_i and α_i are the ith bits of α' and α, respectively). Let $A(\vec{p}, \vec{q}), B(\vec{p}, \vec{r})$ be two collections of formulas in the displayed variables only, where $\vec{p}, \vec{q}, \vec{r}$ are pairwise disjoint sequences of distinct variables (similar to the notation at the beginning of Section 4.5). Assume that there is no assignment that satisfies both $A(\vec{p}, \vec{q})$ and $B(\vec{p}, \vec{r})$. We say that $A(\vec{p}, \vec{q}), B(\vec{p}, \vec{r})$ are *monotone* if one of the following conditions hold:

1. If $\vec{\alpha}$ is an assignment to $\vec{p}$ and $\vec{\beta}$ is an assignment to $\vec{q}$ such that $A(\vec{\alpha}, \vec{\beta}) = 1$, then for any assignment $\vec{\alpha}' \geq \vec{\alpha}$ it holds that $A(\vec{\alpha}', \vec{\beta}) = 1$.

2. If $\vec{\alpha}$ is an assignment to $\vec{p}$ and $\vec{\beta}$ is an assignment to $\vec{r}$ such that $B(\vec{\alpha}, \vec{\beta}) = 1$, then for any assignment $\vec{\alpha}' \leq \vec{\alpha}$ it holds that $B(\vec{\alpha}', \vec{\beta}) = 1$.

Fix a certain proof system $\mathcal{P}$. Recall the definition of the interpolant function (corresponding to a given unsatisfiable $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$; that is, functions for which (4.41) in Section 4.5 hold). Assume that for every monotone $A(\vec{p}, \vec{q}), B(\vec{p}, \vec{r})$ there is a transformation from every $\mathcal{P}$-refutation of $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$ into the corresponding interpolant *monotone* Boolean circuit $C(\vec{p})$ (that is, $C(\vec{p})$ uses only monotone gates[10]) and whose size

[7]Over all inputs $\vec{x}, \vec{y}$ such that $f(\vec{x}) = 1$ and $f(\vec{y}) = 0$.

[8]The measure CC' is basically the same as CC defined earlier.

[9]Using the known $O(\log N)$-depth bounded fan-in Boolean circuits for the threshold functions: the majority functions have $O(\log n)$-depth circuits, and the threshold functions have a constant-depth reduction to the majority functions (cf., Vollmer (1999), Theorem 1.24 and Section 1.41).

is polynomial in the size of the refutation (note that for every monotone $A(\vec{p},\vec{q}), B(\vec{p},\vec{r})$ the corresponding interpolant circuit must compute a monotone function;[11] the interpolant circuit itself, however, might not be monotone, namely, it may use non-monotone gates). In such a case, we say that $\mathcal{P}$ has the *feasible monotone interpolation property*. This means that, if a proof system $\mathcal{P}$ has the feasible monotone interpolation property, then an exponential lower bound on monotone circuits that compute the interpolant function corresponding to $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ implies an exponential-size lower bound on $\mathcal{P}$-refutations of $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$.

Definition 4.5.8 (Feasible monotone interpolation property) *Let $\mathcal{P}$ be a propositional refutation system. Let $A_1(\vec{p},\vec{q}),\dots,A_k(\vec{p},\vec{q})$ and $B_1(\vec{p},\vec{r}),\dots,B_\ell(\vec{p},\vec{r})$ be two collections of formulas with the displayed variables only (where $\vec{p}$ has n variables, $\vec{q}$ has s variables and $\vec{r}$ has t variables), such that* either *(the set of satisfying assignments of) $A_1(\vec{p},\vec{q}),\dots,A_k(\vec{p},\vec{q})$ meet condition 1 above* or *(the set of satisfying assignments of) $B_1(\vec{p},\vec{r}),\dots,B_\ell(\vec{p},\vec{r})$ meet condition 2 above. Assume that for any such $A_1(\vec{p},\vec{q}),\dots,A_k(\vec{p},\vec{q})$ and $B_1(\vec{p},\vec{r}),\dots,B_\ell(\vec{p},\vec{r})$, if there exists a $\mathcal{P}$-refutation for $A_1(\vec{p},\vec{q}) \wedge \cdots \wedge A_k(\vec{p},\vec{q}) \wedge B_1(\vec{p},\vec{r}) \wedge \ldots \wedge B_\ell(\vec{p},\vec{r})$ of size S then there exists a monotone Boolean circuit separating $\mathcal{U}_A$ from $\mathcal{V}_B$ (as defined in Section 4.5.1) of size polynomial in S. In this case we say that $\mathcal{P}$ possesses the* feasible monotone interpolation property.

4.6 Size Lower Bounds

In this section we establish an exponential-size lower bound on $\mathrm{R}^0(\mathrm{lin})$ refutations of the clique-coloring formulas. We shall employ the theorem of Bonet, Pitassi and Raz in Bonet et al. (1997) that provides exponential-size lower bounds for any semantic refutation of the clique-coloring formulas, having low communication complexity in each refutation-line.

First we recall the strong lower bound obtained by Alon and Boppana (Alon and Boppana (1987)) (improving over Razborov (1985); see also Andreev (1985)) for the (monotone) *clique separator* functions, defined as follows (a function $f : \{0,1\}^n \to \{0,1\}$ is called *monotone* if for all $\alpha \in \{0,1\}^n$, $\alpha' \geq \alpha$ implies $f(\alpha') \geq f(\alpha)$):

Definition 4.6.1 (Clique separator) *A monotone boolean function $Q^n_{k,k'}$ is called a* clique separator *if it interprets its inputs as the edges of a graph on n vertices, and outputs 1 on every input representing a k-clique, and 0 on every input representing a complete k'-partite graph (see Section 4.4.3).*

Recall that a *monotone Boolean circuit* is a circuit that uses only monotone Boolean gates (for instance, only the fan-in two gates $\wedge, \vee$).

[10]For instance, a *monotone Boolean circuit* is a circuit that uses only $\wedge, \vee$ gates of fan-in two (see also Section 4.6). In certain cases, the monotone interpolation technique is also applicable for a larger class of circuits, that is, circuits that compute with real numbers and that can use any nondecreasing real functions as gates (this was proved by Pudlák in Pudlák (1997)).

[11]That is, if $\alpha' \geq \alpha$ then $C(\alpha') \geq C(\alpha)$.

Theorem 4.6.2 (Alon and Boppana (1987), Theorem 3.9) *Let k, k' be integers such that $3 \leq k' < k$ and $k\sqrt{k'} \leq n/(8 \log n)$, then every monotone Boolean circuit that computes a clique separator function $Q^n_{k,k'}$ requires size at least*

$$\frac{1}{8}\left(\frac{n}{4k\sqrt{k'}\log n}\right)^{(\sqrt{k'}+1)/2}.$$

For the next theorem, we need a slightly different (and weaker) version of communication complexity, than that in Definition 4.5.4.

Definition 4.6.3 (Worst-case partition communication complexity) *Let X denote n Boolean variables $x_1, \ldots, x_n$, and let S_1, S_2 be a partition of X into two disjoint sets of variables. The communication complexity of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is the number of bits needed to be exchanged by two players, one knowing the values given to the S_1 variables and the other knowing the values given to S_2 variables, in the worst-case, over all possible partitions S_1 and S_2.*

Theorem 4.6.4 (Bonet et al. (1997), Theorem 5) *Every semantic refutation of $\neg\text{CLIQUE}^n_{k,k'}$ (for $k' < k$) with m refutation-lines and where each refutation-line (considered as a the characteristic function of the line) has worst-case partition communication complexity ζ, can be transformed into a monotone circuit of size $m \cdot 2^{3\zeta+1}$ that computes a separating function $Q^n_{k,k'}$.*

In light of Theorem 4.6.2, in order to be able to apply Theorem 4.6.4 to $\text{R}^0(\text{lin})$, and arrive at an exponential-size lower bound for $\text{R}^0(\text{lin})$ refutations of the clique-coloring formulas, it suffices to show that $\text{R}^0(\text{lin})$ proof-lines have logarithmic worst-case partition communication complexity:

Lemma 4.6.5 *Let c, d be two constants, and let D be an $\text{R}_{c,d}(\text{lin})$-line with N variables. Then, the worst-case partition communication complexity of D is at most $O(\log N)$ (where D is identified here with the characteristic function of D).*

Proof. The proof is similar to the proof of Lemma 4.5.7 for solving Task 1 (and the analogous Task 2) in Definition 4.5.4. □

By direct calculations we obtain the following lower bound from Theorems 4.6.2, 4.6.4 and Lemma 4.6.5:

Corollary 4.6.6 *Let k be an integer such that $3 \leq k' = k - 1$ and assume that $\frac{1}{2} \cdot n/(8 \log n) \leq k\sqrt{k} \leq n/(8 \log n)$. Then, for all $\varepsilon < 1/3$, every $\text{R}^0(\text{lin})$ refutation of $\neg\text{CLIQUE}^n_{k,k'}$ is of size at least $2^{\Omega(n^{\varepsilon})}$.*

When considering the parameters of Theorem 4.4.13, we obtain a super-polynomial separation between $\text{R}^0(\text{lin})$ refutations and $\text{R}(\text{lin})$ refutations, as described below.

From Theorems 4.6.2,4.6.4 and Lemma 4.6.5 we have (by direct calculations):

Corollary 4.6.7 *Let* $k = \sqrt{n}$ *and* $k' = (\log n)^2/8 \log\log n$. *Then, every* $\mathrm{R}^0(\mathrm{lin})$ *refutation of* $\neg\mathrm{CLIQUE}^n_{k,k'}$ *has size at least* $n^{\Omega\left(\frac{\log n}{\sqrt{\log\log n}}\right)}$.

By Corollary 4.4.14, R(lin) admits polynomial-size in n refutations of $\neg\mathrm{CLIQUE}^n_{k,k'}$ under the parameters in Corollary 4.6.7. Thus we obtain the following separation result:

Corollary 4.6.8 R(lin) *is super-polynomially stronger than* $\mathrm{R}^0(\mathrm{lin})$.

Comment: Note that we do not need to assume that the coefficients in "R^0(lin) proof-lines" are constants for the lower bound argument. If the coefficients in "R^0(lin) proofs-lines" are only polynomially bounded (in the number of variables) then the same lower bound as in Corollary 4.6.7 also applies. This is because "R^0(lin) proof-lines" in which coefficients are polynomially bounded integers, still have low (that is, logarithmic) worst-case partition communication complexity.

4.7 Relations with Extensions of Cutting Planes

In this section we tie some loose ends by showing that, in full generality, R(lin) polynomially simulates R(CP) with polynomially bounded coefficients, denoted R(CP*). First we define the R(CP*) proof system – introduced in Krajíček (1998) – which is a common extension of resolution and CP* (the latter is cutting planes with polynomially bounded coefficients). The system R(CP*), thus, is essentially resolution operating with disjunctions of linear inequalities (with polynomially bounded integral coefficients) augmented with the cutting planes inference rules.

A linear inequality is written as

$$\vec{a} \cdot \vec{x} \geq a_0 \,, \tag{4.44}$$

where $\vec{a}$ is a vector of integral coefficients $a_1, \ldots, a_n$, $\vec{x}$ is a vector of variables $x_1, \ldots, x_n$, and a_0 is an integer. The *size* of the linear inequality (4.44) is the sum of all $a_0, \ldots, a_n$ written in *unary notation* (this is similar to the size of linear equations in R(lin)). A *disjunction of linear inequalities* is just a disjunction of inequalities of the form in (4.44). The semantics of a disjunction of inequalities is the natural one, that is, a disjunction is true under an assignment of integral values to $\vec{x}$ if and only if at least one of the inequalities is true under the assignment. The *size of a disjunction of linear inequalities* is the total size of all linear inequalities in it. We can also add in the obvious way linear inequalities, that is, if L_1 is the linear inequality $\vec{a} \cdot \vec{x} \geq a_0$ and L_2 is the linear inequality $\vec{b} \cdot \vec{x} \geq b_0$, then $L_1 + L_2$ is the linear inequality $(\vec{a} + \vec{b}) \cdot \vec{x} \geq a_0 + b_0$.

The proof system R(CP*) operates with disjunctions of linear inequalities with integral coefficients (written in *unary notation*), and is defined as follows (our formulation is similar to that in Kojevnikov (2007)):[12]

[12]When we allow coefficients to be written in *binary notation*, instead of unary notation, the resulting proof system is denoted R(CP).

Definition 4.7.1 (R(CP*)) *Let $K := \{K_1, \dots, K_m\}$ be a collection of disjunctions of linear inequalities (whose coefficients are written in unary notation). An* R(CP*)-proof from K of a disjunction of linear inequalities D *is a finite sequence $\pi = (D_1, ..., D_\ell)$ of disjunctions of linear inequalities, such that $D_\ell = D$ and for each $i \in [\ell]$: either $D_i = K_j$ for some $j \in [m]$; or D_i is one of the following* R(CP*)-axioms*:*

1. $x_i \geq 0$, *for any variable x_i;*
2. $-x_i \geq -1$, *for any variable x_i;*
3. $(\vec{a} \cdot \vec{x} \geq a_0) \vee (-\vec{a} \cdot \vec{x} \geq 1 - a_0)$, *where all coefficients (including a_0) are integers;*

or D_i was deduced from previous lines by one of the following R(CP*)-inference rules*:*

1. *Let A, B be two, possibly empty, disjunctions of linear inequalities and let L_1, L_2 be two linear inequalities.*

   From $A \vee L_1$ and $B \vee L_2$ derive $A \vee B \vee (L_1 + L_2)$.
2. *Let L be some linear equation.*

   From a, possibly empty, disjunction of linear equations A derive $A \vee L$.
3. *Let A be a, possibly empty, disjunction of linear equations.*

   From $A \vee (0 \geq 1)$ derive A.
4. *Let c be a non-negative integer.*

   From $(\vec{a} \cdot \vec{x} \geq a_0) \vee A$ derive $(c\vec{a} \cdot \vec{x} \geq ca_0) \vee A$.
5. *Let A be a, possibly empty, disjunction of linear inequalities, and let $c \geq 1$ be an integer.*

   From $(c\vec{a} \cdot \vec{x} \geq a_0) \vee A$ derive $(a \cdot \vec{x} \geq \lceil a_0/c \rceil) \vee A$.

An R(CP*) refutation of *a collection of disjunctions of linear inequalities K is a proof of the empty disjunction from K. The* size *of a proof π in R(CP*) is the total size of all the disjunctions of linear inequalities in π, denoted $|\pi|$.*

In order for R(lin) to simulate R(CP*) proofs, we need to fix the following translation scheme. Every linear inequality L of the form $\vec{a} \cdot \vec{x} \geq a_0$ is translated into the following disjunction, denoted $\widehat{L}$:

$$(\vec{a} \cdot \vec{x} = a_0) \vee (\vec{a} \cdot \vec{x} = a_0 + 1) \vee \cdots \vee (\vec{a} \cdot \vec{x} = a_0 + k) \ , \tag{4.45}$$

where k is such that $a_0 + k$ equals the sum of all positive coefficients in $\vec{a}$, that is, $a_0 + k = \max_{\vec{x} \in \{0,1\}^n} (\vec{a} \cdot \vec{x})$ (in case the sum of all positive coefficients in $\vec{a}$ is less than a_0, then we put $k = 0$). An inequality with no variables of the form $0 \geq a_0$ is translated into $0 = a_0$ in case it is false (that is, in case $0 < a_0$), and into $0 = 0$ in case it is true (that is, in case $0 \geq a_0$). Note that since the coefficients of linear inequalities (and linear equations) are written in *unary notation*, any linear inequality of size s translates into a disjunction of linear equations of size $O(s^2)$. Clearly, every $0,1$ assignment to the variables $\vec{x}$ satisfies L if and only if it satisfies its translation $\widehat{L}$. A disjunction of linear inequalities D is translated into the disjunction of the translations of all the linear inequalities in it, denoted $\widehat{D}$. A collection $K := \{K_1, \dots, K_m\}$ of disjunctions of linear inequalities, is translated into the collection $\left\{\widehat{K}_1, \dots, \widehat{K}_m\right\}$.

Theorem 4.7.2 R(lin) *polynomially-simulates R(CP*). In other words, if π is an R(CP*) proof of a linear inequality D from a collection of disjunctions of linear inequalities $K_1, \dots, K_t$, then there is an* R(lin) *proof of $\widehat{D}$ from $\widehat{K}_1, \dots, \widehat{K}_t$ whose size is polynomial in $|\pi|$.*

Proof. By induction on the number of proof-lines in π.

Base case: Here we only need to show that the axioms of R(CP*) translates into axioms of R(lin), or can be derived with polynomial-size (in the size of the original R(CP*) axiom) R(lin) derivations (from R(lin)'s axioms).

R(CP*) axiom number (1): $x_i \geq 0$ translates into the R(lin) axiom $(x_i = 0) \vee (x_i = 1)$.

R(CP*) axiom number (2): $-x_i \geq -1$, translates into $(-x_i = -1) \vee (-x_i = 0)$. From the Boolean axiom $(x_i = 1) \vee (x_i = 0)$ of R(lin), one can derive with a constant-size R(lin) proof the line $(-x_i = -1) \vee (-x_i = 0)$ (for instance, by subtracting twice each equation in $(x_i = 1) \vee (x_i = 0)$ from itself).

R(CP*) axiom number (3): $(\vec{a} \cdot \vec{x} \geq a_0) \vee (-\vec{a} \cdot \vec{x} \geq 1 - a_0)$. The inequality $(\vec{a} \cdot \vec{x} \geq a_0)$ translates into

$$\bigvee_{b=a_0}^{h} (\vec{a} \cdot \vec{x} = b)\,,$$

where h is the maximal value of $\vec{a} \cdot \vec{x}$ over $0,1$ assignments to $\vec{x}$ (that is, h is just the sum of all positive coefficients in $\vec{a}$). The inequality $(-\vec{a} \cdot \vec{x} \geq 1 - a_0)$ translates into

$$\bigvee_{b=1-a_0}^{f} (-\vec{a} \cdot \vec{x} = b)\,,$$

where f is the maximal value of $-\vec{a} \cdot \vec{x}$ over $0,1$ assignments to $\vec{x}$ (that is, f is just the sum of all negative coefficients in $\vec{a}$). Note that one can always flip the sign of any equation $\vec{a} \cdot \vec{x} = b$ in R(lin). This is done, for instance, by subtracting twice $\vec{a} \cdot \vec{x} = b$ from itself. So overall R(CP*) axiom number (3) translates into

$$\bigvee_{b=a_0}^{h} (\vec{a} \cdot \vec{x} = b) \vee \bigvee_{b=1-a_0}^{f} (-\vec{a} \cdot \vec{x} = b)\,,$$

that can be converted inside R(lin) into

$$\bigvee_{b=-f}^{a_0-1} (\vec{a} \cdot \vec{x} = b) \vee \bigvee_{b=a_0}^{h} (\vec{a} \cdot \vec{x} = b)\,. \tag{4.46}$$

Let $\mathcal{A}' := \{-f, -f+1, \ldots, a_0 - 1, a_0, a_0+1, \ldots, h\}$ and let $\mathcal{A}$ be the set of all possible values that $\vec{a} \cdot \vec{x}$ can get over all possible Boolean assignments to $\vec{x}$. Notice that $\mathcal{A} \subseteq \mathcal{A}'$. By Lemma 4.2.5, for any $\vec{a} \cdot \vec{x}$, there is a polynomial-size (in the size of the linear form $\vec{a} \cdot \vec{x}$) derivation of $\bigvee_{\alpha \in \mathcal{A}} (\vec{a} \cdot \vec{x} = \alpha)$. By using the R(lin) weakening rule we can then derive $\bigvee_{\alpha \in \mathcal{A}'} (\vec{a} \cdot \vec{x} = \alpha)$ which is equal to (4.46).

Induction step: Here we simply need to show how to polynomially simulate inside R(lin) every inference rule application of R(CP*).

Rule (1): Let A, B be two disjunctions of linear inequalities and let L_1, L_2 be two linear inequalities. Assume we already have a R(lin) proofs of $\widehat{A} \vee \widehat{L_1}$ and $\widehat{B} \vee \widehat{L_2}$. We need to derive $\widehat{A} \vee \widehat{B} \vee \widehat{L_1 + L_2}$. Corollary 4.2.4 shows that there is a polynomial-size (in the size of $\widehat{L_1}$ and $\widehat{L_2}$; which is polynomial in the size of L_1 and L_2) derivation of $\widehat{L_1 + L_2}$ from $\widehat{L_1}$ and $\widehat{L_2}$, from which the desired derivation immediately follows.

Rule (2): The simulation of this rule in R(lin) is done using the R(lin) weakening rule.

Rule (3): The simulation of this rule in R(lin) is done using the R(lin) simplification rule (remember that $0 \geq 1$ translates into $0 = 1$ under our translation scheme).

Rule (4): Let c be a non-negative integer. We need to derive $(\widehat{c\vec{a} \cdot \vec{x} \geq ca_0}) \vee \widehat{A}$ from $(\widehat{\vec{a} \cdot \vec{x} \geq a_0}) \vee \widehat{A}$ in R(lin). This amounts only to "adding together" c times the disjunction $(\widehat{\vec{a} \cdot \vec{x} \geq a_0})$ in $(\widehat{\vec{a} \cdot \vec{x} \geq a_0}) \vee \widehat{A}$. This can be achieved by c many applications of Corollary 4.2.4. We omit the details.

Rule (5): We need to derive $(\widehat{\vec{a} \cdot \vec{x} \geq \lceil a_0/c \rceil}) \vee \widehat{A}$, from $(\widehat{c\vec{a} \cdot \vec{x} \geq a_0}) \vee \widehat{A}$. Consider the disjunction of linear equations $(\widehat{c\vec{a} \cdot \vec{x} \geq a_0})$, which can be written as:

$$(c\vec{a} \cdot \vec{x} = a_0) \vee (c\vec{a} \cdot \vec{x} = a_0 + 1) \vee \ldots \vee (c\vec{a} \cdot \vec{x} = a_0 + r)\,, \tag{4.47}$$

where $a_0 + r$ is the maximal value $c\vec{a} \cdot \vec{x}$ can get over $0, 1$ assignments to $\vec{x}$. By Lemma 4.2.5 there is a polynomial-size (in the size of $\vec{a} \cdot \vec{x}$) R(lin) proof of

$$\bigvee_{\alpha \in \mathcal{A}} (\vec{a} \cdot \vec{x} = \alpha)\,, \tag{4.48}$$

where $\mathcal{A}$ is the set of all possible values of $\vec{a} \cdot \vec{x}$ over $0, 1$ assignments to $\vec{x}$.

We now use (4.47) to cut-off from (4.48) all equations $(\vec{a} \cdot \vec{x} = \beta)$ for all $\beta < \lceil a_0/c \rceil$ (this will give us the desired disjunction of linear equations). Consider the equation $(\vec{a} \cdot \vec{x} = \beta)$ in (4.48) for some fixed $\beta < \lceil a_0/c \rceil$. Use the resolution rule of R(lin) to add this equation to itself c times inside (4.48). We thus obtain

$$(c\vec{a} \cdot \vec{x} = c\beta) \vee \bigvee_{\alpha \in \mathcal{A} \setminus \{\beta\}} (\vec{a} \cdot \vec{x} = \alpha)\,. \tag{4.49}$$

Since β is an integer and $\beta < \lceil a_0/c \rceil$, we have $c\beta < a_0$. Thus, the equation $(c\vec{a} \cdot \vec{x} = c\beta)$ does not appear in (4.47). We can then successively resolve $(c\vec{a} \cdot \vec{x} = c\beta)$ in (4.49) with each equation $(c\vec{a} \cdot \vec{x} = a_0), \ldots, (c\vec{a} \cdot \vec{x} = a_0 + r)$ in (4.47). Hence, we arrive at $\bigvee_{\alpha \in \mathcal{A} \setminus \{\beta\}} (\vec{a} \cdot \vec{x} = \alpha)$. Overall, we can cut-off all equations $(\vec{a} \cdot \vec{x} = \beta)$, for $\beta < \lceil a_0/c \rceil$, from (4.48). We then get the disjunction

$$\bigvee_{\alpha \in \mathcal{A}'} (\vec{a} \cdot \vec{x} = \alpha)\,,$$

where $\mathcal{A}'$ is the set of all elements of $\mathcal{A}$ greater or equal to $\lceil a_0/c \rceil$ (in other words, all values greater or equal to $\lceil a_0/c \rceil$ that $\vec{a} \cdot \vec{x}$ can get over $0, 1$ assignments to $\vec{x}$). Using the weakening rule of R(lin) (if necessary) we can arrive finally at the desired disjunction $(\widehat{\vec{a} \cdot \vec{x} \geq \lceil a_0/c \rceil})$, which concludes the R(lin) simulation of R(CP*)'s inference Rule (5). □

4.8 Chapter Summary

In this chapter we explored extensions of resolution of varying strength that operate with disjunctions of linear equations instead of clauses. We demonstrated efficient proofs for counting arguments in resolution over linear equations and its fragments. We showed that already a proper fragment of resolution over linear equations is capable of efficiently proving certain hard formulas. We also obtained exponential lower bounds on fragments of resolution over linear equations using the monotone interpolation technique, and provided a simulation of full cutting planes proofs with polynomially bounded coefficients by resolution over linear equations.

In the next chapter we shall link resolution over linear equations with multilinear proofs. Specifically, we show that depth-3 multilinear proofs can polynomially simulate R^0(lin) proofs.

Chapter 5

From Resolution over Linear Equations to Multilinear Proofs

In this chapter we apply the results about resolution over linear equations from Chapter 4 to obtain new results for multilinear proof systems. In particular, we shall demonstrate a polynomial simulation of R^0(lin) by multilinear proofs operating with depth-3 formulas. This simulation will rely heavily on the fact that multilinear symmetric polynomials have small depth-3 multilinear formulas over fields of characteristic 0 (see Preliminaries, Section 2.5.4).

5.1 From R(lin) Proofs to PCR Proofs

Here we demonstrate a general and straightforward translation from R(lin) proofs into PCR proofs over fields of characteristic 0. We use the term "translation" in order to distinguish it from a *simulation*; since here we are not interested in the size of PCR proofs. In fact we have not defined the size of PCR proofs at all. We shall be interested only in the *number of steps* in PCR proofs.

From now on, all polynomials and arithmetic formulas are considered over some fixed field $\mathbb{F}$ *of characteristic* 0. Recall that any field of characteristic 0 contains (an isomorphic copy of) the integer numbers, and so we can use integer coefficients in the field.

Definition 5.1.1 (Polynomial translation of R(lin) proof-lines) *Let D be a disjunction of linear equations:*

$$\left(a_1^{(1)}x_1 + \ldots + a_n^{(1)}x_n = a_0^{(1)}\right) \vee \cdots \vee \left(a_1^{(t)}x_1 + \ldots + a_n^{(t)}x_n = a_0^{(t)}\right). \tag{5.1}$$

We denote by $\widehat{D}$ its translation into the following polynomial:[1]

$$\left(a_1^{(1)}x_1 + \ldots + a_n^{(1)}x_n - a_0^{(1)}\right) \cdots \left(a_1^{(t)}x_1 + \ldots + a_n^{(t)}x_n - a_0^{(t)}\right). \tag{5.2}$$

If D is the empty disjunction*, we define $\widehat{D}$ to be the polynomial* 1.

It is clear that every $0,1$ assignment to the variables in D, satisfies D, if and only if $\widehat{D}$ evaluates to 0 under the assignment.

Proposition 5.1.2 *Let $\pi = (D_1, \ldots, D_\ell)$ be an* R(lin) *proof sequence of D_ℓ, from some collection of initial disjunctions of linear equations $Q_1, \ldots, Q_m$. Then, there exists a PCR proof of $\widehat{D}_\ell$ from $\widehat{Q}_1, \ldots, \widehat{Q}_m$ with at most a polynomial in $|\pi|$ number of steps.*

Proof. We proceed by induction on the number of lines in π. *The base case* is the translation of the axioms of R(lin) via the translation scheme in Definition 5.1.1. An R(lin) Boolean axiom $(x_i = 0) \vee (x_i = 1)$ is translated into $x_i \cdot (x_i - 1)$ which is already a Boolean axiom of PCR.

[1]This notation should not be confused with the same notation in Section 4.4.3.

For the *induction step*, we translate an R(lin) inference rule application into a polynomial-size PCR proof sequence as follows. We use the following simple claim:

Claim: Let p and q be two polynomials and let s be the minimal size of an arithmetic formula computing q. Then one can derive from p in PCR the product $q \cdot p$, with only a polynomial in s number of steps.

Proof of claim: By induction on s. $\blacksquare_{\text{Claim}}$

Assume that $D_i = D_j \vee L$ was derived from D_j using the weakening inference rule of R(lin), where $j < i \leq \ell$ and L is some linear equation. Then, by Claim 5.1, $\widehat{D}_i = \widehat{D}_j \cdot \widehat{L}$ can be derived from $\widehat{D}_j$ with a derivation of at most polynomial in $|D_j \vee L|$ many steps.

Assume that D_i was derived from D_j where D_j is $D_i \vee (0 = k)$, using the simplification inference rule of R(lin), where $j < i \leq \ell$ and k is a non-zero integer. Then, $\widehat{D}_i$ can be derived from $\widehat{D}_j = \widehat{D}_i \cdot -k$ by multiplying with $-k^{-1}$ (via the Addition rule of PCR).

Thus, it remains to simulate the *resolution rule* application of R(lin). Let A, B be two disjunctions of linear equations and assume that $A \vee B \vee ((\vec{a} - \vec{b}) \cdot \vec{x} = a_0 - b_0)$ was derived in π from $A \vee (\vec{a} \cdot \vec{x} = a_0)$ and $B \vee (\vec{b} \cdot \vec{x} = b_0)$.

We need to derive $\widehat{A} \cdot \widehat{B} \cdot ((\vec{a} - \vec{b}) \cdot \vec{x} - a_0 + b_0)$ from $\widehat{A} \cdot (\vec{a} \cdot \vec{x} - a_0)$ and $\widehat{B} \cdot (\vec{b} \cdot \vec{x} - b_0)$. This is done by multiplying $\widehat{A} \cdot (\vec{a} \cdot \vec{x} - a_0)$ with $\widehat{B}$ and multiplying $\widehat{B} \cdot (\vec{b} \cdot \vec{x} - b_0)$ with $\widehat{A}$ (using Claim 5.1), and then subtracting the resulted polynomials from each other. $\square$

Remark: When translating R(lin) proofs into PCR proofs we actually do not make any use of the "negative" variables $\bar{x}_1, \ldots, \bar{x}_n$. Nevertheless, the multilinear proof systems make use of these variables in order to polynomially simulate PCR proofs (see Theorem 3.3.1 and its proof in Chapter 3 Section 3.3).

We shall need the following corollary in the sequel:

Corollary 5.1.3 *Let c, d be two constants, let $\pi = D_1, \ldots, D_\ell$ be an* R(lin) *proof of D_ℓ using only* $\mathrm{R}_{c,d}$(lin)*-lines, and let s be the maximal size of an* $\mathrm{R}_{c,d}$(lin)*-line in π. Then there are two constants c', d' that depend only on c, d and a PCR proof π' of $\widehat{D}_\ell$ with polynomial-size in $|\pi|$ number of steps, and such that every line of π' has size at most polynomial in s and is a translation (via Definition 5.1.1) of an* $\mathrm{R}_{c',d'}$(lin)*-line.*

Proof. The simulation of R(lin) by PCR shown above, can be thought of as, first, considering $\widehat{D}_1, \ldots, \widehat{D}_\ell$ as the "skeleton" of a PCR proof of $\widehat{D}_\ell$. And second, for each D_i that was deduced by one of R(lin)'s inference rules from previous lines, one inserts the corresponding PCR proof sequence that simulates the appropriate inference rule application (as described in the proof of Proposition 5.1.2). By definition, those PCR proof-lines that correspond to lines in the skeleton $\widehat{D}_1, \ldots, \widehat{D}_\ell$ are translations of $\mathrm{R}_{c,d}$(lin)-lines (with size at most polynomial in s). Thus, to conclude the proof of the corollary, one needs only to check that for any $\mathrm{R}_{c,d}$(lin)-line D_i that was deduced by one of R(lin)'s inference rules from previous $\mathrm{R}_{c,d}$(lin)-lines (as demonstrated in the proof of Proposition 5.1.2), the inserted corresponding PCR proof sequence uses only translations of $\mathrm{R}_{c',d'}$(lin)-lines (with size polynomial in s), for two constants c', d' that depend only on c, d. This can be verified by a straightforward inspection. $\square$

5.2 From PCR Proofs to Multilinear Proofs

First recall the general simulation result (Theorem 3.3.1) proved in Chapter 3. This theorem states the following: Let π be a PCR refutation of some initial collection of multilinear polynomials Q over some fixed field. Assume that π has polynomially many steps (that is, the number of proof lines in the PCR proof sequence is polynomial). If the multilinearization (namely, the result of applying the $\mathrm{M}[\cdot]$ operator [Definition 2.5.2]) of each of the polynomials in π has a polynomial-size depth d multilinear formula (with a plus gate at the root), then there is a polynomial-size depth-d fMC refutation of Q.

5.2.1 Multilinearization of Polynomials

Here we show that multilinear proofs operating with depth-3 multilinear formulas (that is, depth-3 fMC) over fields of characteristic 0 polynomially simulate R^0(lin) proofs. In light of Proposition 5.1.3 and Theorem 3.3.1, to this end it suffices to show that for constants c, d, any $\mathrm{R}_{c,d}$(lin)-line D translates into a corresponding polynomial p (via the translation in Definition 5.1.1) such that $\mathrm{M}[p]$ has a multilinear formula of size polynomial (in the number of variables) and depth at most 3 (with a plus gate at the root) over fields of characteristic 0.

We need the following simple properties (given without a proof):

Proposition 5.2.1 *Fix a field $\mathbb{F}$ and let X be a finite set of variables.*

- *If p, q are two symmetric polynomials over X, then the product $p \cdot q$ is also a symmetric polynomial over X;*
- *If p is a symmetric polynomial over X, then $\mathrm{M}[p]$ is a multilinear symmetric polynomial over X.*

From Theorem 2.5.4 and Proposition 5.2.1 we get:

Corollary 5.2.2 *Let $\mathbb{F}$ be a field of characteristic 0 and let X be a set of ℓ variables. If p is a product of (one or more) symmetric polynomials over X (over the field $\mathbb{F}$), then $\mathrm{M}[p]$ has a depth-3 multilinear formula of size polynomial (in ℓ), with a plus gate at the root.*

We shall also need the following more general proposition, which might be interesting by itself:

Proposition 5.2.3 *Let $\mathbb{F}$ be a field of characteristic 0. For a constant c, let $X_1, \dots, X_c$ be c finite sets of variables (not necessarily disjoint), where $\Sigma_{i=1}^{c}|X_i| = \ell$. Let $f_1, \dots, f_c$ be c symmetric polynomials over $X_1, \dots, X_c$ (over the field $\mathbb{F}$), respectively. Then, there is a depth-3 multilinear formula for $\mathrm{M}[f_1 \cdots f_c]$ of size polynomial (in ℓ), with a plus gate at the root.*

Remark. Note the difference between Corollary 5.2.2 and Proposition 5.2.3. Corollary 5.2.2 speaks about a (finite) *unbounded* product of symmetric polynomials over the *same* set of variables. On the other hand, Proposition 5.2.3 speaks about a product of *constant* number of symmetric polynomials over *different* (but not necessarily disjoint) sets of variables.

Proof. We shall need the following two basic claims (given without a proof).

Claim: Let X be a set of ℓ variables $x_1, \dots, x_\ell$, and let p_1, p_2 be two multilinear polynomials over X such that for all $0,1$ assignments to $x_1, \dots, x_\ell$, $p_1(x_1, \dots, x_\ell) = p_2(x_1, \dots, x_\ell)$. Then $p_1 = p_2$ as *formal polynomials*.

Since symmetric polynomials are invariant under renaming of variables then restricted to $0,1$ assignments the values of symmetric polynomials are determined only by the *number of* 1*'s* in their input variables. Formally, if p is a symmetric polynomial of degree d from $\mathbb{F}[X]$ in ℓ variables, then there is a polynomial h of degree at most d in one variable, such that for $0,1$ assignments to $x_1, \dots, x_\ell$, $p(x_1, \dots, x_\ell) = h(x_1 + \dots + x_\ell)$. Hence, if we let $Y_1, \dots, Y_m$ be pairwise disjoint subsets of $X = \{x_1, \dots, x_\ell\}$, such that $\biguplus_{i=1}^m Y_i = X$, then we have the following:

Claim: Let p be a symmetric polynomial from $\mathbb{F}[X]$ of degree d, then there is a polynomial h of degree at most d in m variables, such that for all $0,1$ assignments to $x_1, \dots, x_\ell$,

$$p(x_1, \dots, x_\ell) = h\left(\sum_{x_i \in Y_1} x_i, \dots, \sum_{x_i \in Y_m} x_i\right).$$

We are now ready to prove Proposition 5.2.3. Let $\mathbf{X} := \bigcup_{i=1}^c X_i$. Let $m := 2^c$ and partition $\mathbf{X}$ into at most m disjoint subsets as follows. For every $J \subseteq [c]$, let $X_J := \bigcap_{i \in J} X_i \setminus \bigcup_{i \in [c] \setminus J} X_i$ and define the abbreviation

$$z_J := \sum_{x_i \in X_J} x_i \,. \tag{5.3}$$

(This way, the variables in X_i are exactly the variables that occur in all z_J, such that $i \in J \subseteq [c]$.) Let $J_1, \dots, J_m$ be all the subsets of $[c]$, and let z_k denote z_{J_k}, for every $1 \le k \le m$.

We clearly have,

$$\mathbf{M}\left[\prod_{i=1}^c f_i\right] = \mathbf{M}\left[\prod_{i=1}^c \mathbf{M}[f_i]\right] . \tag{5.4}$$

By Proposition 5.2.1, for all $1 \le i \le c$, $\mathbf{M}[f_i]$ is a (multilinear) symmetric polynomial. Thus, by Claim 5.2.1, for all $1 \le i \le c$ there exists a polynomial $g_i(y_1, \dots, y_m)$ of degree at most ℓ (with at most m variables), such that $\mathbf{M}[f_i] = g_i(z_1, \dots, z_m)$ for all assignments of $0,1$ values *to the variables in* X_i (note that $g_i(y_1,\ldots,y_m)$ is not necessarily a multilinear polynomial in the y_j's).[2]

Hence, by (5.4) for all assignments of $0,1$ to the variables in $\mathbf{X}$,

$$\mathrm{M}\left[\prod_{i=1}^{c} f_i\right] = \mathrm{M}\left[\prod_{i=1}^{c} g_i(z_1,\ldots,z_m)\right] \tag{5.5}$$

(note that the multilinearization operator $\mathrm{M}[\cdot]$ in the right hand side of (5.5) operates on the polynomial $\prod_i^c g_i(z_1,\ldots,z_m)$ considered as a polynomial in the $\mathbf{X}$ variables).

Therefore, by Claim 5.2.1, the two sides of (5.5) are equal as *formal* polynomials (over $\mathbf{X}$). Since c and m are constants, $\prod_i^c g_i(y_1,\ldots,y_m)$ can be written as a sum of polynomially many monomials in the variables $y_1,\ldots,y_m$. Thus, when substituting z_j's for y_j's (for all $1 \le j \le m$), $\prod_i^c g_i(z_1,\ldots,z_m)$ can be written as a sum of polynomially many products of the form $\prod_{j=1}^m z_j^{e_j}$ (where the e_j's stand for some non-negative integers). Hence, by linearity of $\mathrm{M}[\cdot]$, the right hand side of (5.5) can be written as a sum of polynomially many terms of the form $\mathrm{M}\left[\prod_{j=1}^m z_j^{e_j}\right]$. It remains only to prove the following:

Claim: Every polynomial of the form $\mathrm{M}\left[\prod_{j=1}^m z_j^{e_j}\right]$ (where the e_j's stand for some non-negative integers) has a depth 3 multilinear formula (in the variables in $\mathbf{X}$) of size polynomial in ℓ and a plus gate at the root .

Proof of claim: Since the sets of variables that occur in each of the z_j's are pairwise disjoint, $\mathrm{M}\left[\prod_{j=1}^m z_j^{e_j}\right] = \prod_{j=1}^m \mathrm{M}\left[z_j^{e_j}\right]$. For every $1 \le j \le m$, $z_j^{e_j}$ is a product of symmetric polynomials (in (not necessarily all) the variables in $\mathbf{X}$). Thus, by Corollary 5.2.2, $\mathrm{M}\left[z_j^{e_j}\right]$ can be written as a sum of polynomially (in ℓ) many products of linear polynomials (in other words, a polynomial-size leveled depth 3 multilinear formula with a plus gate at the root). Since m is a constant, $\prod_{j=1}^m \mathrm{M}\left[z_j^{e_j}\right]$ can be written as a sum of polynomially many terms, where each term is a product of (polynomially many) linear polynomials over *disjoint sets of variables*. In other words, we have reached a polynomial-size (in ℓ) depth 3 multilinear formula. $\blacksquare_{\text{Claim}}$

This completes the proof of Proposition 5.2.3. $\square$

Lemma 5.2.4 *Let s,t be two constants that do not depend on n, let D be an* $\mathrm{R}_{s,t}$(lin)*-line with n variables and let $p = \widehat{D}$ (see Definition 5.1.1). Then,* $\mathrm{M}[p]$ *has a depth-*3 *multilinear formula over fields of characteristic* 0*, with a plus gate at the root and size at most polynomial in the size of D.*

Proof. Assume that the underlying variables of D are $\vec{x} = x_1\ldots,x_n$. By assumption, we can partition the disjunction D into a *constant number* t of disjuncts, where one disjunct

[2]For any $1 \le k \le m$, the variable y_k actually occurs in g_i if and only if $i \in J_k$. The other variables y_k are still indicated for ease of writing.

is a (possibly empty, translation of a) clause C, and all other disjuncts have the following form:

$$\bigvee_{i=1}^{m} (\vec{a} \cdot \vec{x} = \ell_i) \ , \tag{5.6}$$

where the ℓ_i's are integers, m is not necessarily bounded and $\vec{a}$ denotes a vector of n *constant* integer coefficients, each having absolute value at most s.

Suppose that the clause C is $\bigvee_{i\in\mathcal{I}} x_i \vee \bigvee_{j\in\mathcal{J}} \neg x_j$, and let us denote by $q = \prod_{i\in\mathcal{I}}(x_i - 1) \cdot \prod_{j\in\mathcal{J}} x_j$ the polynomial representing C.

Consider a disjunct as shown in (5.6). Since the coefficients $\vec{a}$ are constants (having absolute value at most s), $\vec{a} \cdot \vec{x}$ can be written as a sum of constant number of linear forms, each with the *same* constant coefficient. In other words, $\vec{a} \cdot \vec{x}$ can be written as $z_1 + \ldots + z_d$, for some constant d, where for all $i \in [d]$:

$$z_i := b \cdot \sum_{j\in J} x_j \ , \tag{5.7}$$

for some $J \subseteq [n]$ and some constant integer b. We shall assume without loss of generality that d is the same constant for every disjunct of the form (5.6) inside D (otherwise, take d to be the maximal such d).

Thus, (5.6) is translated (via the translation scheme in Definition 5.1.1) into:

$$\prod_{i=1}^{m} (z_1 + \ldots + z_d - \ell_i) \, . \tag{5.8}$$

By fully expanding the product in (5.8), we arrive at:

$$\sum_{r_1+\ldots+r_{d+1}=m} \left(\alpha_{r_{d+1}} \cdot \prod_{k=1}^{d} z_k^{r_k} \right) , \tag{5.9}$$

where the r_i's are non-negative integers, and where the α_r's, for every $0 \le r \le m$ are just integer coefficients, formally defined as follows (this definition is not essential; we present it only for the sake of concreteness):

$$\alpha_r := \sum_{\substack{U\subseteq[m]\\ |U|=r}} \prod_{j\in U} (-\ell_j) \, . \tag{5.10}$$

Claim: The polynomial $\widehat{D}$ (the polynomial translation of D) is a linear combination (over $\mathbb{F}$) of polynomially (in $|D|$) many terms, such that each term can be written as

$$q \cdot \prod_{k\in K} z_k^{r_k} \ ,$$

where K is a collection of a constant number of indices, r_k's are non-negative integers, and the z_k's and q are as above (that is, the z_k's are linear forms, where each z_k has a single coefficient for all variables in it, as in (5.7), and q is a polynomial translation of a clause).

Proof of claim: By assumption, the total number of disjuncts of the form (5.6) in D is $\leq t$. Consider the polynomial (5.9) above. In $\widehat{D}$, we actually need to multiply at most t many polynomials of the form shown in (5.9) and the polynomial q.

For every $j \in [t]$ we write the (single) linear form in the jth disjunct as a sum of constantly many linear forms $z_{j,1} + \ldots + z_{j,d}$, where each linear form $z_{j,k}$ has the same coefficient for every variable in it. Thus, $\widehat{D}$ can be written as:

$$q \cdot \prod_{j=1}^{t} \left(\sum_{r_1+\ldots+r_{d+1}=m_j} \underbrace{\left(\alpha^{(j)}_{r_{d+1}} \cdot \prod_{k=1}^{d} z_{j,k}^{r_k} \right)}_{(\star)} \right), \tag{5.11}$$

(where the m_j's are not bounded, and the coefficients $\alpha^{(j)}_{r_{d+1}}$ are as defined in (5.10) except that here we add the index (j) to denote that they depend on the jth disjunct in D). Denote the maximal m_j, for all $j \in [t]$, by m_0. The size of D, denoted $|D|$, is at least m_0. Note that since d is a constant, the number of summands in each (middle) sum in (5.11) is polynomial in m_0, which is at most polynomial in $|D|$. Thus, by expanding the outermost product in (5.11), we arrive at a sum of polynomially in $|D|$ many summands. Each summand in this sum is a product of t terms of the form designated by $(\star)$ in Equation (5.11), multiplied by q. $\blacksquare_{\text{Claim}}$

It remains to apply the multilinearization operator (Definition 2.5.2) on $\widehat{D}$, and verify that the resulting polynomial has a depth-3 multilinear formula with a plus gate at the root and of polynomial-size (in $|D|$). Since $\mathbf{M}[\cdot]$ is a linear operator, it suffices to show that when applying $\mathbf{M}[\cdot]$ on each summand in $\widehat{D}$, as described in Claim 5.2.1, one obtains a (multilinear) polynomial that has a depth-3 multilinear formula with a plus gate at the root, and of polynomial-size in the number of variables n (note that clearly $n \leq |D|$). This is established in the following claim:

Claim: The polynomial $\mathbf{M}\big[q \cdot \prod_{k\in K} z_k^{r_k}\big]$ has a depth-3 multilinear formula of polynomial-size in n (the overall number of variables) and with a plus gate at the root (over fields of characteristic 0), under the same notation as in Claim 5.2.1.

Proof of claim: Recall that a power of a symmetric polynomial is a symmetric polynomial in itself. Since each z_k (for all $k \in K$) is a symmetric polynomial, then its power $z_k^{r_k}$ is also symmetric. The polynomial q is a translation of a clause, hence it is a product of two symmetric polynomials: the symmetric polynomial that is the translation of the disjunction of literals with positive signs, and the symmetric polynomial that is the translation of the disjunction of literals with negative signs. Therefore, $q \cdot \prod_{k\in K} z_k^{r_k}$ is a product of constant number of symmetric polynomials. By Proposition 5.2.3 $\mathbf{M}\big[q \cdot \prod_{k\in K} z_k^{r_k}\big]$ (where here the $\mathbf{M}[\cdot]$ operator operates on the $\vec{x}$ variables in the z_k's and q) is a polynomial for which there is a polynomial-size (in n) depth-3 multilinear formula with a plus gate at the root (over fields of characteristic 0). $\blacksquare_{\text{Claim}}$ $\square$

We now come to the main result of this chapter.

Corollary 5.2.5 *Multilinear proofs operating with depth-*3 *multilinear formulas (that is, depth-*3 fMC *proofs) polynomially-simulate* $R^0(\text{lin})$ *proofs.*

Proof. Immediate from Corollary 5.1.3, Theorem 3.3.1 and Proposition 5.2.4.

For the sake of clarity we repeat the chain of transformations needed to prove the simulation. Let π be a given $R^0(\text{lin})$ proof that uses only $R_{c,d}(\text{lin})$-lines, for c, d two constant integers. We first use Corollary 5.1.3 to transform π into a PCR proof π', with number of steps that is at most polynomial in $|\pi|$, and where each line in π' is a polynomial translation of some $R_{c,d}(\text{lin})$-line with size at most polynomial in the maximal line in π (which is clearly at most polynomial in $|\pi|$). Thus, by Proposition 5.2.4 each polynomial in π' has a corresponding multilinear polynomial with a polynomial-size in $|\pi|$ depth-3 multilinear formula (and a plus gate at the root). Therefore, by Theorem 3.3.1, we can transform π' into a depth-3 fMC proof with only a polynomial (in $|\pi|$) increase in size. □

5.3 Applications to Multilinear Proofs: Small Depth-3 Multilinear Proofs

5.3.0.1 The Pigeonhole Principle

Recall the propositional formulation of the pigeonhole principle described in Chapter 4 Section 4.4.1. The two sets of clauses in (4.24) translate via Definition 3.0.7 to the following set of polynomials:

$$\begin{array}{ll} \text{Pigeons}: & \forall i \in [m], \;\; \bar{x}_{i,1} \cdots \bar{x}_{i,n} \\ \text{Holes}: & \forall i < j \in [m]\, \forall k \in [n], \;\; x_{i,k} \cdot x_{j,k} \end{array} \tag{5.12}$$

By Theorem 4.4.4, $R^0(\text{lin})$ admits polynomial-size (in n) refutations of the m to n pigeonhole principle (for any $m > n$) (as defined in Definition 4.4.1). Thus, Corollary 5.2.5 yields:

Theorem 5.3.1 *For any $m > n$ there are polynomial-size (in n) depth-*3 fMC *refutations of the m to n pigeonhole principle* PHP^m_n *(over fields of characteristic* 0*).*

Haken (1985) showed an exponential lower bound on the size of resolution refutations of the pigeonhole principle (where the number of holes is n and the number of pigeons is $n + 1$). Pitassi et al. (1993) and, independently, Krajíček et al. (1995) showed exponential lower bounds on the size of proofs of the pigeonhole principle in bounded-depth Frege (again, where the number of holes is n and number of pigeons is $n + 1$). Razborov (1998) and, subsequently, Impagliazzo et al. (1999) showed exponential lower bounds on the size (and degree) of PC-refutations of a different *low degree* version of the pigeonhole principle. In this low degree version the Pigeon polynomials of (5.12) are replaced by $1 - (x_{i,1} + \ldots + x_{i,n})$, for all $i \in [m]$. This low degree version is not a translation of a CNF formula. Our upper bound is also applicable to this low-degree version of the pigeonhole principle (however, this would not yield a separation of fMC from PC and PCR as we consider these proof systems as proof systems for (polynomial translations of) CNF formulas).

Grigoriev and Hirsch (2003) showed polynomial size refutations of the pigeonhole principle (i.e., polynomials (5.12)) for a formal (syntactic) propositional proof system denoted $\mathcal{F}\text{-}\mathcal{PC}$ (see the discussion on the fMC proof system in Chapter 3 and Chapter 6 for more on syntactic algebraic proof systems) that manipulates general arithmetic formulas. Their refutations operate with general arithmetic formulas of constant-depth.

By the discussion above we have:

Corollary 5.3.2 *Depth-*3 fMC *over fields of characteristic* 0 *has an exponential gap over resolution and bounded-depth Frege for the pigeonhole principle.*

Since depth-3 fMC polynomially simulates resolution (Proposition 3.2.2) then by Corollary 5.3.2 depth-3 fMC is strictly stronger than resolution.

5.3.0.2 Tseitin mod p Formulas

Corollary 5.2.5 and Theorem 4.4.9 yield:

Theorem 5.3.3 *Let G be an r-regular graph with n vertices, where r is a constant, and fix some modulus p. Then there are polynomial-size (in n) depth-*3 fMC *refutations of Tseitin* mod p *formulas* $\mathrm{BTS}_{G,p}$ *(over fields of characteristic* 0*).*

The polynomial-size refutations of Tseitin graph tautologies here are different than those demonstrated in Chapter 3 (Section 3.5). Theorem 5.3.3 establishes polynomial-size refutations over any field of characteristic 0 of Tseitin mod p formulas, whereas the former proofs required the field to contain a primitive pth root of unity. On the other hand, the refutations in Section 3.5 of Tseitin mod p formulas do not make any use of the semantic nature of the fMC proof system, in the sense that they do not utilize the fact that the base field is of characteristic 0. (The latter fact enables one to efficiently represent any symmetric [multilinear] polynomial by a depth-3 multilinear formula.)

5.4 Chapter Summary

In this chapter we linked together the results of the previous two chapters (Chapters 3, 4). Specifically, we showed that depth-3 multilinear proofs can already polynomial simulate a certain considerably strong fragment of R(lin) (namely, R^0(lin)). As a consequence we obtained polynomial upper bounds on depth-3 multilinear proofs of the propositional pigeonhole principle and the Tseitin's mod p formulas (the latter is different from the short proofs demonstrated in Chapter 3).

Chapter 6

Symbolic Proofs of Polynomial Identities: From Semantic to Syntactic Algebraic Proofs

This chapter aims at developing further the theory of algebraic proof complexity. Here, instead of establishing propositional tautologies, we shall consider algebraic proof systems *establishing that a given arithmetic formula computes the zero polynomial* (or, equivalently, that two given arithmetic formulas compute the same polynomial). We begin with the basic definition of the model and then we continue (in Section 6.5) with the problem of proving lower bounds.

6.1 Preliminaries

For a graph G we write $|G|$ to denote the number of vertices in G. For an edge directed from u to v in G, we also say that u *points to* v.

Assumptions on arithmetic formulas in this chapter. In this chapter it will be convenient to consider all arithmetic formulas as labeled trees **with fan-in at most two** and where **field elements can only label leaves, and not edges** (compare this to the definition of arithmetic formulas in Section 2.5.1). An arithmetic with a plus or product gate at the root is said to be a *plus formula* or *product formula*, respectively. Given an arithmetic formula Φ a *subformula* of Φ is any (not necessarily proper) subtree of Φ. We say that *an arithmetic formula* φ *occurs in an arithmetic formula* φ' if φ is a subformula of φ'. In this case we also say that φ' *contains* φ as a subformula.

We write $\Phi_1 \equiv \Phi_2$ if Φ_1 and Φ_2 are two syntactically equal formulas (equal as labeled trees; not to be confused with equality between polynomials). For a formula Φ and a node v in Φ, we write Φ_v to denote the subformula of Φ whose root is v. We write $\Phi\{\psi\}_v$ to denote the formula Φ where the subtree rooted by v in Φ is replaced[1] by ψ. We write $\Phi\{\psi\}$ (without explicitly displaying v) to mean that ψ is a subformula of Φ.

Constant-depth arithmetic formulas. We shall consider bounded-depth formulas. Since we deal in this chapter with formulas with fan-in two, *a depth-d formula* will mean that there is an a priori constant d that bounds the number of alternations between plus and product gates in every path in the formula-graph (given a path p from the root to a leaf, the number of alternations between plus and product gates is the number of alternations between consecutive blocks of the same gate-labels). A formula Φ is said to be a $\Sigma\Pi\Sigma\ldots$ *formula* (where $\Sigma\Pi\Sigma\ldots$ has $d \geq 1$ symbols) if every path in Φ starting at the root and ending in the immediate ancestor of a leaf in the formula-graph of Φ is labeled with a block of (zero or more) consecutive plus gates followed by a block of (zero or more) consecutive product gates and so forth (d times). If moreover, a $\Sigma\Pi\Sigma\ldots$ (with $d \geq 1$ symbols) formula Φ contains a path starting at the root and ending in the immediate ancestor of a leaf that is labeled with a block of *one or more* consecutive plus gates followed by a block of *one or more* consecutive product gates and so forth (d times), then we say that Φ is a *proper* $\Sigma\Pi\Sigma\ldots$ *formula*. The definition of (proper) $\Pi\Sigma\Pi\ldots$ *formulas* is similar.

[1]Note that Φ_v is identified here with the labeled tree rooted in v, and not with all the formulas that are equivalent to the labeled tree rooted in v. In other words, when we replace Φ_v by ψ in Φ, we only replace the subtree of Φ whose root is v, by the tree ψ.

Comment: When considering depth-3 formulas we shall slightly change the definition of depth (see Section 6.2.1) in order to conform to the standard definition of depth-3 arithmetic formulas (that is, as a sum of products of linear forms).

Notational conventions. In this chapter we deal exclusively with arithmetic formulas, and so we will often use the term "formula" to mean an arithmetic formula. Recall that given two formulas Φ_1 and Φ_2, we write $\Phi_1 + \Phi_2$ and $\Phi_1 \times \Phi_2$ to designate the formulas with a plus (respectively, product) gate at the root and Φ_1, Φ_2 as its two children. We will also use parenthesis to designate explicitly the structure of formulas. For example, $(x_1 + x_2) + (x_3 + x_4)$ means that $(x_1 + x_2)$ and $(x_3 + x_4)$ are the two subformulas attached to the root gate (while, $(x_2 + x_3)$, for instance, is not a subformula of the main formula). Most often we will not care for the associativity and order of multiplication and addition, to the effect that we shall not write explicitly the parentheses in formulas, like in $\Phi_1 \times \Phi_2 \times \Phi_3$. Further, we write $\prod_{i \in I} \Phi_i$, where I is some set of indices, to mean the product formula whose products are all Φ_i (for $i \in I$), where we ignore the associativity of subformulas (formally, every product gate in this formula is still of fun-in 2; though this is not essential). Similarly, we will write $\sum_{i \in I} \Phi_i$ for the plus formula of all Φ_i ($i \in I$). Also, we will sometimes abuse notation by identifying arithmetic formulas with the polynomials they compute.

6.2 Analytic Symbolic Proofs of Polynomial Identities

Let us fix our underlying field $\mathbb{F}$. Unless otherwise stated, from this point on, all formulas will be arithmetic formulas computing polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. An arithmetic formula computes the zero polynomial if the polynomial computed at the root of the formula is the zero polynomial (e.g., the formula $x_1 + (-1 \times x_1)$). In this section we describe our underlying proof system, that is, *analytic symbolic proof systems for polynomial identities*. The system introduced here is complete and sound for the set of arithmetic formulas computing the zero polynomial. In other words, every formula computing the zero polynomial has a proof (completeness), and only formulas computing the zero polynomial have proofs (soundness).

Definition 6.2.1 (Derivation rule) *A* derivation rule *is a pair of formulas F, G, written as:*

$$(\star)\,\frac{F}{G}\,,$$

where $(\star)$ is the name *of the derivation rule. Let Φ be a formula and v a node in Φ. Assume that $\Phi_v \equiv F$ (that is, $\Phi \equiv \Phi\{F\}_v$). Then, given the formula Φ and the derivation rule $(\star)$, we can derive from Φ the formula $\Phi\{G\}_v$, in which case we say that $\Phi\{G\}_v$* was derived from Φ by the derivation rule $(\star)$ applied on v.

Notation: Let Φ be a formula and v a node in Φ, such that $\Phi_v \equiv F$, and suppose that $\Phi\{G\}_v$ was derived from Φ by the derivation rule $(\star)$ applied on v. Then we will say that the derivation rule was *applied in* or *on* ψ, in case ψ is a subformula of Φ that contains v. Further, in this case we call the formula G *the consequence of the application of rule* $(\star)$. We write

$$\frac{\Phi\{F\}_v}{\Phi\{G\}_v}\,(\star)\,,$$

to denote the above derivation rule application, where $(\star)$ designates the name of the rule that was applied in order to derive the formula in the lower-line from the formula in the upper-line. (It should be clear from the context that this latter notation is meant to denote a proof sequence [see Definition 6.2.3 below], and not the description of a derivation rule.)

The following definition formulates the standard polynomial-ring axioms, where "non-analytic" rules are kept out (see discussion below).

Definition 6.2.2 (Polynomial-ring analytic derivation rules) *The following rules are the* polynomial-ring analytic derivation rules *(where Q_1, Q_2, Q_3 range over all arithmetic formulas computing polynomials in $\mathbb{F}[x_1, \ldots, x_n]$):*

Zero element rules:

$$\frac{0 \times Q_1}{0} \qquad \frac{0 + Q_1}{Q_1}$$

Unit element rules:

$$\frac{1 \times Q_1}{Q_1} \qquad \frac{Q_1}{1 \times Q_1}$$

Scalar rules: let $\alpha, \alpha_1, \alpha_2$ be elements in $\mathbb{F}$.

$$\frac{\alpha_1 + \alpha_2}{\alpha}\ (\text{where } \alpha = \alpha_1 + \alpha_2)$$

$$\frac{\alpha_1 \times \alpha_2}{\alpha}\ (\text{where } \alpha = \alpha_1 \cdot \alpha_2)$$

$$\frac{\alpha}{\alpha_1 \times \alpha_2}\ (\text{where } \alpha = \alpha_1 \cdot \alpha_2)$$

Commutativity:

$$\frac{Q_1 + Q_2}{Q_2 + Q_1} \qquad \frac{Q_1 \times Q_2}{Q_2 \times Q_1}$$

Associativity:

$$\frac{Q_1 + (Q_2 + Q_3)}{(Q_1 + Q_2) + Q_3} \qquad \frac{Q_1 \times (Q_2 \times Q_3)}{(Q_1 \times Q_2) \times Q_3}$$

Forward distributivity:

$$\frac{Q_1 \times (Q_2 + Q_3)}{(Q_1 \times Q_2) + (Q_1 \times Q_3)}$$

Backward distributivity:

$$\frac{(Q_1 \times Q_2) + (Q_1 \times Q_3)}{Q_1 \times (Q_2 + Q_3)}$$

Comment: It is easy to show that by using the commutativity and associativity (to the left) rules in Definition 6.2.2, one can efficiently simulate the associativity to the right rules.

Definition 6.2.3 (Analytic symbolic proofs of polynomial identities) *Let Φ and Φ' be two arithmetic formulas (computing the same polynomial in $\mathbb{F}[x_1,\ldots,x_n]$). An* analytic symbolic derivation of Φ' from Φ *is a sequence of arithmetic formulas $\psi_1,\ldots,\psi_m$ such that $\psi_1 \equiv \Phi$, $\psi_m \equiv \Phi'$, and for all $1 < i \leq m$, ψ_i is derived from ψ_{i-1} by applying the polynomial-ring analytic derivation-rules (applicable on any subformula) from Definition 6.2.2. If the initial formula Φ computes the zero polynomial and Φ' is the formula 0, then we call such a derivation of Φ' from Φ an* analytic symbolic proof of Φ.

The ***length*** *of an analytic symbolic proof (or derivation)* is defined to be the number of proof-lines in the proof (derivation, respectively).

We shall prove the completeness (and soundness) of analytic symbolic proofs in Section 6.4.

Discussion about analytic symbolic proofs and the subformula property. Let us explain our choice of derivation rules in Definition 6.2.2. Recall from the introductory chapter (Section 1.2.3.3) that we aim at formulating *analytic* symbolic proofs, that is, a system that enjoys a sort of subformula property. The intuitive interpretation of this property in our setting would be to prevent the prover from using "clever tricks" (or "detours") when proving polynomial identities, in the sense that one cannot introduce new algebraic formulas that might help in efficiently proving the identities (like, introducing new monomials or new formulas, later to be cut-off in the proof).

The subformula property usually states that every formula that appears in an analytic proof sequence π of T also appears in the terminal proof-line T. In our setting this should mean that the consequence (i.e., lower-line) in any rule may only contain subformulas already appearing in the premise (i.e., upper-line) of the rule. (Note that in a standard sequent calculus proof, the proof starts with the axioms and terminates in a tautology; this should be analogous, in our setting, to a symbolic proofs of an arithmetic formula computing the zero polynomial *taken backward*: one starts from the "axiom" formula 0 and develops the formula computing the zero-polynomial; thus, whereas the subformula criterion usually requires that every formula in an upper-line of a rule occurs as a subformula in the lower-line of the rule, we should require that every (sub)formula in a lower-line of a rule should appear in some sense in the upper-line of the rule.) However, in our system we cannot follow precisely this requirement, since the two distributivity rules might change the structure of subformulas (for instance, $(Q_1 \times Q_2)$ in the lower-line of the forward distributivity rule is not a subformula of the upper-line $Q_1 \times (Q_2 + Q_3)$). Nevertheless, analytic symbolic proofs keep a relaxed subformula property (a "sub-monomial property", so to speak), as we now explain.

We say that a monomial is *syntactically computed by an arithmetic formula* Φ if it occurs in the set of monomials that results when expanding all the monomials in Φ while using *no canceling of monomials* (no canceling of monomials occurs also in any gate of Φ, not just the root gate).[2] In analytic symbolic proofs we have the following "sub-monomial property": If π is an analytic symbolic proof and the formula Φ is a proof-line (*not a proper* subformula in a proof-line) in π, then every monomial that is syntactically computed by Φ either is syntactically computed by the initial proof-line (again, not a proper subformula of the initial proof-line), or is the sum of (two or more) monomials that are syntactically computed in the initial proof-line. This way, *the number of monomials syntactically computed by each proof-line does not increase along the proof sequence*. We shall not prove this statement formally here, and so this idea should only be kept as an intuition in the mind of the reader.

Consider the following three (sound) derivation rules (*absent* from our system):

$$\text{(i)}\ \frac{Q_1}{Q_1+0}\ ;\ \text{(ii)}\ \frac{0}{0\times Q_1}\ ;\ \text{(iii)}\ \frac{\alpha}{\alpha_1+\alpha_2}\quad \text{(where } \alpha=\alpha_1+\alpha_2, \text{ for } \alpha_1,\alpha_2,\alpha\in\mathbb{F}\text{)}.$$

We explain now how the choice not to include the above three rules helps us in keeping the relaxed form of the subformula property in our proof system. First, given Φ, one cannot simply derive $\Phi+\Delta-\Delta$ from Φ, where Δ is any non-empty formula. Note that for this derivation to be possible, we would need the following derivation sequence that uses rules (i) and (ii) above:

$$\begin{array}{cl}
\Phi & \\ \hline
\Phi+0 & \text{apply rule (i)} \\ \hline
\Phi+0\times\Delta & \text{apply rule (ii)} \\ \hline
\Phi+(1-1)\times\Delta & \\ \hline
\Phi+1\times\Delta+(-1)\times\Delta & \\ \hline
\Phi+\Delta+(-1)\times\Delta\ . &
\end{array}$$

Second, if we had the rule (iii) in our proof system it would be possible to add arbitrary number of new monomials that are syntactically computed by proof-lines throughout the proof, as the following example illustrates:

$$\begin{array}{cl}
3\times(x_1\times x_2) & \\ \hline
(2+1)\times(x_1\times x_2) & \text{apply rule (iii)} \\ \hline
2\times(x_1\times x_2)+1\times(x_1\times x_2) & \\ \hline
\cdots &
\end{array}$$

6.2.1 Depth-3 formulas and depth-3 analytic symbolic proofs

We now consider analytic symbolic proofs of polynomial identities operating with depth-3 arithmetic formulas. The standard definition of depth-3 (and specifically $\Sigma\Pi\Sigma$) arithmetic formulas includes all formulas that can be written as a sum of products of linear polynomials. In other words, according to the standard definition, a $\Sigma\Pi\Sigma$ formula Φ (in the

[2] A monomial here means a product of variables *with its scalar coefficient*.

variables $x_1, \ldots, x_n$) can be written as:

$$\Phi \equiv \sum_{i=1}^{m} \prod_{j=1}^{d_i} L_{ij}\,, \tag{6.1}$$

where the L_{ij}'s are linear polynomials in the variables $x_1, \ldots, x_n$.

Due to the syntactic nature of our symbolic proof system, we require that a field element $\alpha \in \mathbb{F}$ that multiplies a (polynomial computed by a) formula f is written as $\alpha \times f$, where the product gate $\times$ is written explicitly. This makes, in our setting, the polynomial in (6.1) to be a depth-4 formula, that is a $\Sigma\Pi\Sigma\Pi$ (the reason is that variables inside a linear polynomial L_{ij} might have coefficients, which makes L_{ij} a $\Sigma\Pi$ formula). In order to include polynomials of the form shown in Equation (6.1) in our depth-3 proof systems we define the following class of formulas.

Definition 6.2.4 ($\hat{\Sigma}$ and $\Sigma\Pi\hat{\Sigma}$ formulas) *A $\hat{\Sigma}$ formula is a $\Sigma\Pi\Sigma$ arithmetic formula (according to the definitions in the beginning of this chapter), such that the bottom $\Pi\Sigma$ level may include only field elements or products of a single variable with a sum (of zero or more) field elements. (Accordingly a $\Sigma\Pi\hat{\Sigma}$ formula is a $\Sigma\Pi\Sigma\Pi\Sigma$ arithmetic formula where the bottom $\Sigma\Pi\Sigma$ level is $\hat{\Sigma}$.)*

Example: The formula $(2+4+1) \times x_1 + 3 \times x_2 + (1+2) \times x_3 + 1$, is a $\hat{\Sigma}$ formula. The formula 3×2 is *not* a $\hat{\Sigma}$ formula.

Thus, a $\hat{\Sigma}$ formula is a formula computing a linear form (we need to include sums of fields elements as coefficients and not just a single field element as a coefficient, since this will enable us to add two linear forms using only $\hat{\Sigma}$ formulas). Note that indeed any sum of products of linear polynomials can be computed by a $\Sigma\Pi\hat{\Sigma}$ formula.

We conclude that when dealing with depth-3 proof systems we will in fact assume that all formulas in the proofs are $\Sigma\Pi\hat{\Sigma}$ formulas.

Definition 6.2.5 (Depth-3 analytic symbolic proofs) *A* depth-3 analytic symbolic derivation (proof) *is an analytic symbolic derivation (proof, respectively) in which every proof-line is a $\Sigma\Pi\hat{\Sigma}$ formula.*

We shall prove the completeness of depth-3 analytic symbolic proofs in Section 6.4 (this is done by proving the completeness of a fragment of depth-3 analytic symbolic proofs).

Example: A typical application of the backward distributivity rule inside depth-3 symbolic proofs proceeds according to the following scheme:

$$\frac{\Delta + \prod_{j=1}^{d} L_j \times L' + \prod_{j=1}^{d} L_j \times L''}{\Delta + \prod_{j=1}^{d} L_j \times (L' + L'')}\,,$$

where L', L'' and the L_j's are formulas of linear polynomials in the variables $x_1, \ldots, x_n$ and Δ is any possibly empty formula (note that the two occurrences of $\prod_{j=1}^{d} L_j$ in the upper-line must be *identical*).

(When exchanging the upper- and lower- lines, we get a typical application of the *forward* distributivity rule.)

6.3 The Structure of Symbolic Proofs

In this section we develop terminology and concepts for dealing with structural properties of symbolic proofs. The notions developed here are suitable mainly for dealing with small depth symbolic proofs, as the notions mainly take into account the top gates of the formulas in the proofs. This will suffice for stating and proving our main lower bounds.

We will identify a certain graph structure induced by symbolic proofs. The idea is to partition the graph into levels, each level corresponds to a proof-line. Each node in level i corresponds to one summand of the formula in the ith line of the proof. From each summand (that is, a vertex in the graph) there will be edge(s) directed to its "immediate ancestor(s)" in the previous line. The formal definition follows.

Definition 6.3.1 (Underlying graph of analytic symbolic proof) *Let $\pi = (\Phi_1, \ldots, \Phi_m)$ be an analytic symbolic proof. Define the* underlying directed acyclic graph of π *denoted G_π as follows. The graph G_π has m levels and edges may only go from level i to level $i-1$, for $1 < i \le m$ (the vertices in level 1 have no outgoing edges). For any $1 \le i \le m$, write Φ_i as $\varphi_1 + \ldots + \varphi_{\ell_i}$, for some $\ell_i \ge 1$, where every φ_j is either a product formula or a formula containing only a single leaf. Then, the ith level of G_π consists of ℓ_i vertices, each vertex is labeled by a different summand φ_j from Φ_i and if $1 \le i < m$ then the incoming edges of level i (from vertices in level $i+1$) are defined as follows (we shall sometime abuse notation by identifying a vertex in the graph G_π with its label and a level in the graph G_π with its corresponding proof-line in π):*

(i) In case no rule was applied on φ_j in level i, for $j \in [\ell_i]$ (see notation after Definition 6.2.1), then φ_j appears (as a single vertex) in both level i and level $i+1$, and we put an incoming edge from φ_j in level $i+1$ to φ_j in level i.

(ii) Assume that a rule different from the forward distributivity rule was applied on φ_j in level i, for $j \in [\ell_i]$. Since φ_j is in a separate vertex in G_π and some rule was applied on φ_j (in the ith step in π) then it must be that φ_j is a product formula (and not a single leaf formula). It can be verified by a straightforward inspection of the derivation rules (Definition 6.2.2) that a consequence of any rule different from the forward distributivity rule is not a plus formulas, and so the consequence of φ_j in proof-line $i+1$ in π is (still) a product formula. This implies that there is a single vertex φ'_j in level $i+1$ which is the consequence of φ_j. We define φ_j in level i to have a single incoming edged from φ'_j in level $i+1$.

(iii) *In case the forward distributivity rule was applied on φ_j in level i, for $j \in [\ell_i]$, on the root gate of φ_j (again, see notation after Definition 6.2.1), then the consequence of φ_j in level $i+1$ is a sum of two formulas denoted $\psi_0 + \psi_1$. Thus (by definition of the vertices in G_π), level $i+1$ contains two vertices ψ_0 and ψ_1, and we define φ_j to have two incoming edges, one from ψ_0 and one from ψ_1.*

(iv) *In case the forward distributivity rule was applied in φ_j in level i, for $j \in [\ell_i]$, but not on the root gate of φ_j, then the consequence of φ_j in level $i+1$ must be a* product formula *denoted φ'_j. We define φ_j in level i to have a single incoming edged from φ'_j in level $i+1$.*

(v) *In case the backward distributivity rule was applied on φ_j and φ_{j+1} in level i, for $j \in [\ell_i - 1]$ (note that the backward distributivity rule must be applied on a plus formula, and so it must involve two vertices in level i of G_π), then the consequence of $\varphi_j + \varphi_{j+1}$ in level $i+1$ is a product formula denoted ψ. We define φ_j to have an incoming edge from ψ in level $i+1$.*

Notation: For a vertex v in G_π we denote by $\dot{v}$ the formula that labels v and by $\mathsf{level}(v)$ the level of G_π in which v occurs.

Example. The following derivation:

$$\begin{array}{c}
((x_1 \times x_2) + (2 \times (x_1 + (x_1 + 3)) + (-1 \times (2 \times (x_1 + 3) + x_1 \times (x_2 + 2)))) \\ \hline
((x_1 \times x_2) + (2 \times x_1) + (2 \times (x_1 + 3)) + (-1 \times (2 \times (x_1 + 3) + x_1 \times (x_2 + 2)))) \\ \hline
(x_1 \times x_2) + (2 \times x_1) + (2 \times (x_1 + 3)) + (-1 \times (2 \times (x_1 + 3))) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times x_2) + (2 \times x_1) + (2 \times (x_1 + 3)) + ((-1 \times 2) \times (x_1 + 3)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times x_2) + (2 \times x_1) + (2 \times (x_1 + 3)) + (-2 \times (x_1 + 3)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times x_2) + (x_1 \times 2) + (2 \times (x_1 + 3)) + (-2 \times (x_1 + 3)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times (x_2 + 2)) + (2 \times (x_1 + 3)) + (-2 \times (x_1 + 3)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times (x_2 + 2)) + ((2 - 2) \times (x_1 + 3)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times (x_2 + 2)) + (0 \times (x_1 + 3)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(x_1 \times (x_2 + 2)) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
(1 \times (x_1 \times (x_2 + 2))) + (-1 \times (x_1 \times (x_2 + 2))) \\ \hline
((1 - 1) \times (x_1 \times (x_2 + 2))) \\ \hline
(0 \times (x_1 \times (x_2 + 2))) \\ \hline
0
\end{array}$$

has the corresponding graph structure shown in Figure 6.1 (we ignore the associativity rule applications).

6.3.1 Regular Analytic Symbolic Proofs

We define here a fragment of analytic symbolic proofs, called *regular analytic symbolic proofs*, which mimics to some extent a tree-like structure on analytic symbolic proofs. We

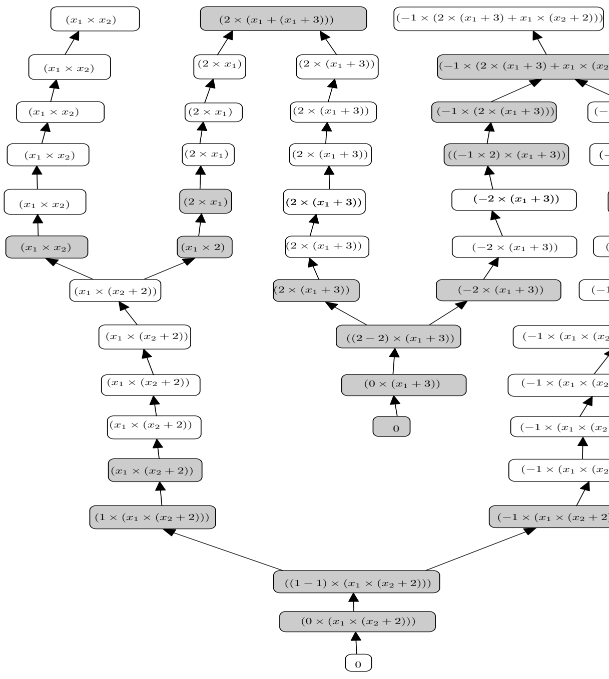
$(x_1 \times x_2)$
$(2 \times (x_1 + (x_1 + 3)))$
$(-1 \times (2 \times (x_1 + 3) + x_1 \times (x_2 + 2)))$
$(x_1 \times x_2)$
$(2 \times x_1)$
$(2 \times (x_1 + 3))$
$(-1 \times (2 \times (x_1 + 3) + x_1 \times (x_2 + 2)))$
$(x_1 \times x_2)$
$(2 \times x_1)$
$(2 \times (x_1 + 3))$
$(-1 \times (2 \times (x_1 + 3)))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times x_2)$
$(2 \times x_1)$
$(2 \times (x_1 + 3))$
$((-1 \times 2) \times (x_1 + 3))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times x_2)$
$(2 \times x_1)$
$(2 \times (x_1 + 3))$
$(-2 \times (x_1 + 3))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times x_2)$
$(x_1 \times 2)$
$(2 \times (x_1 + 3))$
$(-2 \times (x_1 + 3))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times (x_2 + 2))$
$(2 \times (x_1 + 3))$
$(-2 \times (x_1 + 3))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times (x_2 + 2))$
$((2 - 2) \times (x_1 + 3))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times (x_2 + 2))$
$(0 \times (x_1 + 3))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times (x_2 + 2))$
0
$(-1 \times (x_1 \times (x_2 + 2)))$
$(x_1 \times (x_2 + 2))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$(1 \times (x_1 \times (x_2 + 2)))$
$(-1 \times (x_1 \times (x_2 + 2)))$
$((1 - 1) \times (x_1 \times (x_2 + 2)))$
$(0 \times (x_1 \times (x_2 + 2)))$
0

Figure 6.1: Underlying graph of an analytic symbolic proof (this proof is also regular [see Definition 6.3.4]). A single shaded vertex in a level means that a derivation rule is applied on this vertex (that is, an application rule is applied on one of the gates [formally, nodes] in the formula labeling the vertex). Two shaded vertices in a level means that the backward distributivity rule is applied in π on the plus gate (in the corresponding proof-line in π) that has the two shaded vertices as its children.

shall use the following two definitions for that purpose.

Atomic formulas are essentially formulas computing single monomials:

Definition 6.3.2 (Atomic and non-atomic formulas) *A formula Φ is* atomic *if it has the form $\phi_1 \times \cdots \times \phi_k$, for $k \geq 1$, where each ϕ_i ($1 \leq i \leq k$) is either a variable or a sum of one or more field elements; otherwise, Φ is said to be* non-atomic.

Example: The formulas $(1 + 2) \times x_2 \times x_1$ and $1 \times 3 \times x_1$ are atomic formulas, as well as the formula 0 and the formula x_1. The formula $x_2 + 3$ is a non-atomic formula.

Definition 6.3.3 *Given a proof π and its underlying graph G_π let G' be the subgraph of G_π induced by considering only the non-atomic vertices in G_π, and let v be a vertex in G'. Then, $\mathbb{T}_v(G_\pi)$ is defined to be the* directed *subgraph of G' induced by all the vertices (in G') reachable from v via a directed-path in G' (including v itself).*

The idea of the regularity condition of analytic symbolic proofs we are about to define is to guarantee that if a forward distribution rule is applied on $A \times (B + C)$, which breaks the formula into the sum of two terms $A \times B$ and $A \times C$, then the two formulas $A \times B$ and

$A \times C$, as well as any other two formulas that originate from $A \times B$ and $A \times C$ (possibly also originating by other formulas), might not be united together again by means of the backward distributivity rule (in fact, this rule is the only rule that can "unite" two separate summands into a product formula) .

In the terminology of the graph structure of analytic symbolic proofs (Definition 6.3.1) the regularity condition means that a vertex v in a proof-graph cannot have two edge-disjoint (directed) paths starting in v and leading to the same vertex.

Definition 6.3.4 (Regular analytic symbolic proofs) *Let π be an analytic symbolic proof. We say that π is a* regular analytic symbolic proof *(or* regular proof *for short), if for every (non-atomic) vertex v in G_π, the subgraph $\mathbb{T}_v(G_\pi)$ is a* tree.[3]

In other words, analytic symbolic proofs are regular if for every (non-atomic) vertex w in their underlying proof-graph there are no two distinct directed paths originating in w that reach a common vertex (different from w).

Example: Figure 6.1 illustrates a regular analytic symbolic proof. One can verify that for every (non-atomic) vertex w in the graph there are no two distinct directed paths originating in w that reach a common vertex.

Comment: In the definition of regular proofs we need to specifically refer to non-atomic vertices (through the definition of $\mathbb{T}_v(G_\pi)$) for the following reason. If for *no* vertex v in G_π does one get by two distinct (directed) paths from v to a vertex corresponding to an initial summand (in the initial formula), then some formulas computing the zero polynomial might not be provable. Consider for example the formula $(x_1 + 1) \times (x_1 - 1) - x_1 \times x_1 - 1$. This formula computes the zero polynomial. However, the first term (from the left) is a product formula that when expanded, contains the two canceling monomials $x_1 \times -1$ and $1 \times x_1$. Thus, in order to reach the formula 0 we must enable these two terms to cancel each other (that is, to derive the (sub-)formula 0 from them). Hence, this amounts to a vertex labeled $((1 - 1) \times x_1)$ in G_π from which two distinct paths lead to the same initial summand $(x_1 + 1) \times (x_1 - 1)$.

We use the following simple structural proposition in the next sections. Essentially, the proposition states that if there is a derivation starting with $\Delta + \Theta$ and terminates in $\Delta' + \varphi$, and φ in the terminal line was "originated" only from Θ in the initial line, then there is a derivation starting in Θ and terminating in $\Delta'' + \varphi$ for some possibly empty formula Δ''.

Proposition 6.3.5 *Let Θ and Δ be two formulas, and assume that φ is a product formula (or a single leaf formula) such that there is a regular analytic symbolic derivation π starting from $\Delta + \Theta$ and deriving $\Delta' + \varphi$, where Δ' is any possibly empty formula. Denote by G_π the underlying graph of π, and let v be the vertex in the maximal level of G_π that is labeled with φ. Let $\mathbf{T}$ be the set of vertices of $\mathbb{T}_v(G_\pi)$, let $\mathbf{A}$ be the set of all the vertices*

[3]The intention here, of course, is that $\mathbb{T}_v(G_\pi)$ is a tree also when considered as an undirected graph, that is, when replacing every directed edge in $\mathbb{T}_v(G_\pi)$ with an undirected one.

in level 1 *of* G_π*, and let* $\mathbf{B}$ *be the set of all the vertices that are labeled with summands of* Θ *in the first level of* G_π*. If* $(\mathbf{T} \cap \mathbf{A}) \subseteq \mathbf{B}$*, then there is a regular analytic symbolic derivation* π' starting from Θ alone *and deriving* $\Delta'' + \varphi$*, where* Δ'' *is any possibly empty formula.*

Proof. Starting from $\mathbb{T}_v(G_\pi)$, we construct a new graph G' that is a legitimate underlying graph of a regular analytic symbolic proof of $\Delta'' + \varphi$, where Δ'' is any possibly empty formula (we ignore the order of additions).

Let $G' := \mathbb{T}_v(G_\pi)$.

Step I: For every $u \in \mathbf{B} \setminus \mathbf{T}$, we add to every level $1 \leq j \leq \mathsf{level}(v)$ in[4] G' a new copy of u, and put an edge from u in level j to u in level $(j-1)$ in G' (for $1 < j \leq \mathsf{level}(v)$).

Step II: For all vertices w in $\mathbb{T}_v(G_\pi)$ such that $w \neq v$, we do the following. Let $\ell = \mathsf{level}(w)$. By definition of $\mathbb{T}_v(G_\pi)$ there is a vertex $u \in \mathbf{T}$ such that w has an incoming edge from u in $\mathbb{T}_v(G_\pi)$ (since there is a directed path from v to w in $\mathbb{T}_v(G_\pi)$). Assume that there is a vertex $r \in G_\pi \setminus \mathbf{T}$ such that w has an incoming edge from r. Then,

(i) we add the vertex r to level $\ell + 1$ in G'; and

(ii) we add an edge from r to w in G'; and

(iii) we add to every level $j > \ell + 1$ in G' a new copy of r, and put an edge from r in level j to r in level $(j-1)$ in G'.

We claim that G' is the underlying graph of a regular analytic symbolic derivation starting from Θ and deriving $\Delta'' + \varphi$, for some possibly empty formula Δ''. Note that the first level of G' consists precisely of the summands of Θ, and that the last level of G' contains the vertex v. Moreover, G' conforms to the regularity conditions: no vertex in G' has two distinct (non trivial) directed paths that reach the same vertex in G'.

Thus, we only need to show that G' constitutes a legal analytic symbolic derivation. To this end, it suffices to show that every level $j > 1$ in G' corresponds to a proof-line that is derivable from the previous level $j-1$, and that the edges from vertices in level j to the vertices in level $j-1$ in G' are legal (that is, conform to the definition of an underlying graph of an analytic symbolic proof):

Let w be a vertex in level $j > 1$ in G', and consider the following cases:

1. If $w \notin \mathbf{T}$ and w was added in Step I to G', then w was added to G' along with a single outgoing edge that points into another copy of w in level $j-1$ in G', and so we are done.

2. If $w \in \mathbf{T}$, then, by definition, there is an edge from w to another vertex $u \in \mathbf{T}$ and hence u is in G'. In case u has fan-in 1 in G_π we are done.

   Otherwise, there is a (unique) vertex $r \neq w$ in G_π such that r has an edge e pointing to u in G_π. By the construction of G' (Step II, items (i) and (ii) above), both r and the edge e are in G', and so both w and r and their outgoing edges correspond to a legal derivation. This takes care of all the vertices in level j in G' that are not in $\mathbf{T}$, *as well as all the vertices* r *added to* G' *in level* j*, in Step II, item (i) above.*

[4]Clearly, every vertex in $\mathbb{T}_v(G_\pi)$ retain the same level it had in G_π.

3. If $w \notin \mathbf{T}$ and w was added to G' in Step II, item (iii) above, then w has a single outgoing edge that points to a copy of w in level $j-1$ of G', and we are done.

□

Comment: By inspecting the proof of Proposition 6.3.5, it is evident that the statement of this proposition also holds for regular analytic symbolic proofs *operating with depth-*3 *formulas*. We shall use this fact in the sequel.

6.4 Completeness

Recall that when dealing with depth-3 formulas computing the zero polynomial, we assume that the formulas are in fact $\Sigma\Pi\hat{\Sigma}$ formulas (Definition 6.2.4).

Theorem 6.4.1 (Completeness of regular analytic depth-3 symbolic proofs) *Regular analytic depth-*3 *symbolic proofs are* complete *in the sense that every depth-*3 *formula computing the zero polynomial has such a proof; and* sound *in the sense that only formulas computing the zero polynomial have such proofs.*

Proof sketch: Soundness stems from the soundness of the derivation rules: every derivation rule that is applied on some formula Φ yields a formula computing the same polynomial as Φ. We turn to proving completeness.

We first expand all the (formulas of) monomials (including their coefficients, and with no cancelations applied anywhere) in each summand in the initial $\Sigma\Pi\hat{\Sigma}$ formula. This step requires *only the successive application of the forward distributivity rule*. In particular, assume that $\Phi \equiv \psi_1 + \ldots + \psi_k$, for some $k \geq 1$, is a $\Sigma\Pi\hat{\Sigma}$ formula computing the zero polynomial, where each ψ_i is a $\Pi\hat{\Sigma}$ formula. Then, applying the forward distributivity rule on the root product gate of a summand ψ_i (for $i \in [k]$), *yields a new* $\Sigma\Pi\hat{\Sigma}$ *formula*.

Next we derive the 0 formula from every group of canceling monomials. Thus, we arrive at a sum of 0's formulas. This can be further reduced into a single 0 by the zero element rule.

By simple inspection of the underlying graph of the above proof it is easy to see that the graph obtained is *regular*: Every non-atomic vertex has only a single (directed) path that ends at the vertex labeled with a summand from the initial line. Thus, every non-atomic vertex v trivially induces a tree, via $\mathbb{T}_v(G_\pi)$, in the underlying graph, which precisely means that the proof is regular. ■

In the same manner we can get the completeness and soundness properties for unrestricted depth regular symbolic proofs for the set of all formulas that compute the zero polynomial.

Theorem 6.4.2 *Analytic symbolic proofs are sound and complete for the set of formulas computing the zero polynomial.*

Proof. Soundness stems from the soundness of the derivation rules. For completeness, we first expand all the (formulas of) monomials (including their coefficients, and with no cancelations applied anywhere) in each summand in the initial formula (and without taking care for the depth of formulas). Then we proceed as in the proof of Theorem 6.4.1. □

6.5 Lower Bounds

In this section we demonstrate exponential-size lower bounds on the length of regular analytic symbolic proofs operating with depth-3 formulas of certain polynomial identities. The hard instances will be built from small depth-3 formulas that we already met in previous chapters, that is, formulas for the elementary symmetric polynomials over large enough fields.

6.5.1 Hard Formulas

We shall consider from now on all arithmetic formulas to compute polynomials over the field of complex numbers $\mathbb{C}$. The following depth-3 formula will serve as our hard instance.

$$\begin{aligned}\mathsf{Sym}_n :\equiv r_0 \times (x_1+b_0) \times (x_2+b_0) \times \cdots \times (x_n+b_0) \quad &+ \\ r_1 \times (x_1+b_1) \times (x_2+b_1) \times \cdots \times (x_n+b_1) \quad &+ \\ \cdots \qquad & \\ r_n \times (x_1+b_n) \times (x_2+b_n) \times \cdots \times (x_n+b_n) \quad &-1\,,\end{aligned} \tag{6.2}$$

where $r_0, \ldots, r_n$ are some $n+1$ complex numbers and $b_0, b_1, \ldots, b_n$ are $n+1$ *distinct* non-zero complex numbers (each of which is different from 1).

The jth summand in Sym_n, for $0 \le j \le n$, is a $\Pi\Sigma$ formula (and hence specifically a $\Pi\hat{\Sigma}$ formula) and is denoted $A_{n,j}$. In other words,

$$A_{n,j} \equiv r_j \times (x_1+b_j) \times (x_2+b_j) \times \ldots \times (x_n+b_j)\,, \tag{6.3}$$

and so

$$\mathsf{Sym}_n \equiv \sum_{j=0}^{n} A_{n,j} \; -1\,.$$

Proposition 6.5.1 *There exist complex numbers $r_0, \ldots, r_n$ and $n+1$ distinct nonzero complex numbers $b_0, \ldots, b_n$ (each of which is different from 1), so that Sym_n computes the zero polynomial.*

Proof. Consider the polynomial 1 as the symmetric multilinear polynomial of degree 0. By Theorem 2.5.4 in the preliminaries (Chapter 2, Section 2.5.4), there is a depth-3 multilinear arithmetic formula Φ with a plus gate at the root computing 1 where Φ has a special form: specifically, by Equation 2.4 in the proof of Theorem 2.5.4, this formula Φ can be written as:

$$\sum_{j=0}^{n} r_j \cdot \prod_{i=1}^{n} (x_i + b_j) \, ,$$

for any chosen distinct b_j's complex numbers. (Also notice that there must be at least one nonzero r_i $(0 \leq i \leq n)$.) $\square$

6.5.2 Key Lemma: Lower Bounds on Deriving One Formula from Another

In this subsection we show a lower bound on the length of regular analytic symbolic derivations operating with $\Sigma\Pi\hat{\Sigma}$ formulas needed to derive a certain formula from another formula. Specifically, we provide two formulas Φ and Ψ, such that starting from Φ one cannot efficiently derive any formula that *contains* Ψ *as a subformula*. This will then facilitate us in the next subsection (when proving lower bounds on the length of proofs of formulas computing the zero polynomial, that is, on the derivation length needed to reach the formula 0). Note that the task of deriving a certain formula from a different formula, and the task of proving that a formula computes the zero polynomial (by deriving the zero formula) are not necessarily identical tasks since our derivation rules are asymmetric (Definition 6.2.2).[5]

We work in depth-3 from now on, and specifically with $\Sigma\Pi\hat{\Sigma}$ formulas.

Definition 6.5.2 (Derivable subformulas) *Let Φ be a formula. The collection of all formulas Ψ for which there is an analytic symbolic derivation of Ψ from Φ are said to be the formulas* derivable from Φ. *If B is the set of all formulas derivable from Φ then the set of all* subformulas *of formulas in B is called the set of* subformulas derivable from Φ, *denoted* $\mathsf{deriv}(\Phi)$.

Example: Let Φ be the Σ formula $(x_i + b_1)$. Then, the derivable formulas from Φ are, e.g., $(x_i + b_1)$, $(b_1 + x_i)$ and all $\hat{\Sigma}$ formulas $\prod_{i \in I} c_j \times \left(\prod_{k \in K} a_j \times x_i + \prod_{j \in J} b_j \right)$, where $\prod_{i \in I} c_i = \prod_{k \in K} a_k = 1$ and $\prod_{j \in J} b_j = b_1$. Note that $(x_i + b_2 + b_3)$ is *not* derivable from $(x_i + b_1)$, even when $b_2 + b_3 = b_1$ is true in the field; this stems from the definition of our derivation rules (Definition 6.2.2). Further, x_i, b_1 and $(x_i + b_1)$, for instance, are in

[5]To see this, observe that deriving ψ_1 from ψ_2 means that we can derive $\psi_1 - \psi_1$ from $\psi_2 - \psi_1$, and thus we can derive 0 from $\psi_2 - \psi_1$. On the other hand, deriving 0 from $\psi_2 - \psi_1$ does not imply that we can derive ψ_1 from ψ_2 (note that we cannot start from ψ_2, add $-\psi_1 + \psi_1$ to yield $\psi_2 - \psi_1 + \psi_1$, and then derive 0 from the first two summands to yield ψ_1; this is because we would need a rule to introduce the formula $-\psi_1 + \psi_1$ in the first step, and we do not have such a rule in analytic symbolic proofs).

$\mathsf{derive}(x_i + b_1)$, while $(x_i + b_0) \notin \mathsf{derive}(x_i + b_1)$ (in case $b_0 \neq b_1$) (see Proposition 6.5.9 for a proof).

The following definition is basically the transitive closure of a formula under "forward" and "backward" applications of all rules excluding the two distributivity rules (unless the distributivity rules are applied inside $\hat{\Sigma}$ formulas).

Definition 6.5.3 (Simple descendants and simple ancestors) *Given a formula Ψ we define $\mathsf{Cl}(\Psi)$ as the smallest set of formulas containing Ψ and satisfying the following: Assume that Δ is a formula and v is some node in the (tree of) Δ, then:*

(i) If $\Delta\{\phi + 0\}_v \in \mathsf{Cl}(\Psi)$, then $\Delta\{\phi\}_v \in \mathsf{Cl}(\Psi)$;

(ii) If $\Delta\{\alpha_1 + \alpha_2\}_v \in \mathsf{Cl}(\Psi)$, then $\Delta\{\alpha\}_v \in \mathsf{Cl}(\Psi)$ for $\alpha, \alpha_1, \alpha_2 \in \mathbb{F}$, such that $\alpha = \alpha_1 + \alpha_2$;

(iii) $\Delta\{\phi \times 1\}_v \in \mathsf{Cl}(\Psi)$ iff $\Delta\{\phi\}_v \in \mathsf{Cl}(\Psi)$;

(iv) $\Delta\{\alpha_1 \times \alpha_2\}_v \in \mathsf{Cl}(\Psi)$ iff $\Delta\{\alpha\}_v \in \mathsf{Cl}(\Psi)$, for $\alpha, \alpha_1, \alpha_2 \in \mathbb{F}$ such that $\alpha = \alpha_1 \times \alpha_2$;

(v) $\Delta\{\phi_2 \circ \phi_1\}_v \in \mathsf{Cl}(\Psi)$ iff $\Delta\{\phi_1 \circ \phi_2\}_v \in \mathsf{Cl}(\Psi)$, where $\circ \in \{+, \times\}$;

(vi) $\Delta\{(\phi_1 \circ \phi_2) \circ \phi_3\}_v \in \mathsf{Cl}(\Psi)$ iff $\Delta\{\phi_1 \circ (\phi_2 \circ \phi_3)\}_v \in \mathsf{Cl}(\Psi)$, where $\circ \in \{+, \times\}$;

(vii) $\Delta\left\{\left(\sum_{j\in J}\alpha_j + \sum_{k\in K}\alpha_k\right) \times x_i\right\}_v \in \mathsf{Cl}(\Psi)$ (where $|J|, |K| \geq 1$ and for all $j \in J$ and $k \in K$, $\alpha_j, \alpha_k \in \mathbb{F}$) iff $\Delta\left\{\left(\sum_{j\in J}\alpha_j\right) \times x_i + \left(\sum_{k\in K}\alpha_k\right) \times x_i\right\}_v \in \mathsf{Cl}(\Psi)$;

(viii) $\Delta\left\{\left(\sum_{j\in J}\alpha_j\right) \times (x_i + x_j)\right\}_v \in \mathsf{Cl}(\Psi)$ iff $\Delta\left\{\left(\sum_{j\in J}\alpha_j\right) \times x_i + \left(\sum_{j\in J}\alpha_j\right) \times x_j\right\}_v$, where $\alpha_j \in \mathbb{F}$ (for all $j \in J$) and $|J| \geq 1$.

Whenever $\Psi' \in \mathsf{Cl}(\Psi)$ we call Ψ' a simple descendant of *Ψ, and Ψ a* simple ancestor of *Ψ'. Given a formula Ψ we denote by $\mathsf{Cl}^-(\Psi)$ the set of all simple ancestors of Ψ, that is, the set of all Φ such that $\Psi \in \mathsf{Cl}(\Phi)$.*

Note: Observe that $\mathsf{Cl}^-(\Psi)$ is closed under the specified derivation rules when they are applied "*backward*" on the formulas in $\mathsf{Cl}^-(\Psi)$. Also note that the only items in Definition 6.5.3 that are asymmetric are items (i) and (ii).

Comment: The last two clauses (vii) and (viii) in Definition 6.5.3 intend to deal with distributivity rules applied inside $\hat{\Sigma}$ formulas only (and those distributivity rule applications whose consequence is a $\hat{\Sigma}$ formula).

We shall use the following abbreviation (this is identical to Sym_n when excluding the first summand $A_{n,0}$).

$$\begin{aligned} \mathsf{Init}_n :\equiv r_1 \times (x_1 + b_1) \times (x_2 + b_1) \times \cdots \times (x_n + b_1) + \\ r_2 \times (x_1 + b_2) \times (x_2 + b_2) \times \cdots \times (x_n + b_2) + \\ \cdots \\ r_n \times (x_1 + b_n) \times (x_2 + b_n) \times \cdots \times (x_n + b_n) \\ -1\,. \end{aligned} \tag{6.4}$$

We thus have, by Equation (6.3):

$$\mathsf{Init}_n \equiv \sum_{j=1}^{n} A_{n,j} - 1\,. \tag{6.5}$$

Definition 6.5.4 (Proper $\hat{\Sigma}$ formulas) *A proper $\hat{\Sigma}$ formula is a $\hat{\Sigma}$ formula which is a plus formula (that is, it has a plus gate at the root). A proper $\Pi\hat{\Sigma}$ formula is a proper $\Pi\Sigma$ formula where the Σ formulas in the bottom levels are replaced by proper $\hat{\Sigma}$ formulas. Similarly, a proper $\Sigma\Pi\hat{\Sigma}$ formula is a proper $\Sigma\Pi\Sigma$ formula where the Σ formulas in the bottom levels are replaced by proper $\hat{\Sigma}$ formulas.*

Note that every proper Σ formula (Section 6.1) is a proper $\hat{\Sigma}$ formula, while not every proper $\hat{\Sigma}$ formula is a Σ formula.

Lemma 6.5.5 (Key lower bound) *Let π be a regular analytic symbolic proof operating with $\Sigma\Pi\hat{\Sigma}$ formulas, and with the initial formula in π being Init_n, for some positive $n \in \mathbb{N}$. Let G_π be the corresponding graph of π. Assume that v is a vertex in G_π labeled with Φ, which is a simple ancestor of*

$$\psi \times \prod_{k=1}^{m} \Psi_k\,,$$

where ψ is any possibly empty formula, and for every $k \in [m]$, $i \in [n]$ and $j \in [n]$, Ψ_k is a proper $\hat{\Sigma}$ formula, such that $\Psi_k \notin \mathsf{deriv}(x_i + b_j)$. Then $|\mathbb{T}_v(G_\pi)| \geq 2^m$.

Proof. We go by induction on m. The idea is to build inductively and in a bottom-up fashion, starting with v, the tree $\mathbb{T}_v(G_\pi)$, by considering all the possible rules that can derive the formula Φ.

Base case: $m = 1$. The formula Φ is a simple ancestor of $\psi \times \Psi_1$, where ψ is some possibly empty formula[6] and Ψ_1 is a *proper* $\hat{\Sigma}$ formula that is not a subformula derivable from $(x_i + b_j)$, for all $j \in [n]$ and all $i \in [n]$. Observe that (for any positive $n \in \mathbb{N}$) every proper $\hat{\Sigma}$ formula that occurs in Init_n has the form $(x_i + b_j)$, for some $j \in [n]$ and $i \in [n]$.

[6]Note that in this case (i.e., when $m = 1$), since Ψ_1 is a proper $\hat{\Sigma}$ formula, ψ is in fact a non-empty formula, as by definition of G_π, the vertex v cannot be labeled with a plus formula.

Thus, Φ is different from every subformula in the initial formula Init_n, and so the number of proof-lines in π is at least 2, and we are done.

Induction case: We have that Φ is a simple ancestor of $\psi \times \prod_{k=1}^{m} \Psi_k$, where ψ is some possibly empty formula, $m > 1$ and the Ψ_k's are proper $\hat{\Sigma}$ formulas. We shall use the following main technical lemma (whose proof is given in Section 6.5.4).

Lemma 6.5.6 (Technical lemma) *If $m > 1$ then (under the conditions stated in Lemma 6.5.5) there exists a vertex y in $\mathbb{T}_v(G_\pi)$, such that there is a path from v to y in $\mathbb{T}_v(G_\pi)$, and y has two outgoing edges to two (distinct) vertices u, w, so that: u and w are labeled with two simple ancestors of the following product formulas*

$$\psi_0 \times \prod_{k=1}^{m-1} \Psi'_k \qquad \textit{and} \qquad \psi_1 \times \prod_{k=1}^{m-1} \Psi''_k\,, \tag{6.6}$$

respectively, where ψ_0, ψ_1 are some possibly empty formulas, and for all $k \in [m-1]$, $i \in [n]$ and $j \in [n]$, Ψ'_k, Ψ''_k are some proper $\hat{\Sigma}$ formulas such that $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$ and $\Psi''_k \notin \mathsf{deriv}(x_i + b_j)$.

Given Lemma 6.5.6, we can then use the induction hypothesis on the two vertices u, w (whose existence is guaranteed by the lemma), showing that $|\mathbb{T}_u(G_\pi)| \geq 2^{k-1}$ and $|\mathbb{T}_w(G_\pi)| \geq 2^{k-1}$. By the regularity condition we know that $\mathbb{T}_u(G_\pi)$ and $\mathbb{T}_w(G_\pi)$ have no common vertices (note that $\dot{v}$ is a non-atomic formula [Definition 6.3.2] and so indeed $\mathbb{T}_v(G_\pi)$ is a tree [Definition 6.3.4]). Therefore, $|\mathbb{T}_v(G_\pi)| \geq |\mathbb{T}_u(G_\pi)| + |\mathbb{T}_w(G_\pi)| \geq 2^k$, which concludes the proof of Lemma 6.5.5. □

In the next section we shall need the following simple generalization of Lemma 6.5.5:

Corollary 6.5.7 *Let π be a regular symbolic proof operating with $\Sigma\Pi\hat{\Sigma}$ formulas, and with the initial formula in π being $\Delta + \mathsf{Init}_n$, for some $\Sigma\Pi\hat{\Sigma}$ formula Δ and some positive $n \in \mathbb{N}$. Let G_π be the corresponding graph of π. Assume that v is a vertex in G_π labeled with Φ, which is a simple ancestor of*

$$\psi \times \prod_{k=1}^{m} \Psi_k\,,$$

where ψ is any possibly empty formula, and for every $k \in [m]$, $i \in [n]$ and $j \in [n]$, Ψ_k is a proper $\hat{\Sigma}$ formula, such that $\Psi_k \notin \mathsf{deriv}(x_i + b_j)$. Let $\mathbf{T}$ be the set of vertices of $\mathbb{T}_v(G_\pi)$ and let $\mathbf{A}$ be the set of all vertices in level 1 of G_π that are subformulas in Δ (i.e., those corresponding to summands in Δ in the initial line). If $\mathbf{T} \cap \mathbf{A} = \emptyset$, then $|\mathbb{T}_v(G_\pi)| \geq 2^m$.

Proof. Immediate from Lemma 6.5.5 and Proposition 6.3.5. □

6.5.3 Main Lower Bound: Regular Depth-3 Proofs

In this subsection we demonstrate how to use the Key Lemma 6.5.5 in order to prove a lower bound on the *proof* length of Sym_n, that is, a lower bound on the size of derivation starting with Sym_n and terminating with the formula 0.

The main result of this chapter is the following:

Theorem 6.5.8 *Every regular analytic symbolic proof operating with $\Sigma\Pi\hat{\Sigma}$ formulas of* Sym_n *has length* $2^{\Omega(n)}$.

Comment: The number of variables in Sym_n is $\ell = n \cdot (n+1)$, and so Theorem 6.5.8 gives a lower bound of $2^{\Omega(\sqrt{\ell})}$.

The rest of this subsection is dedicated to the proof of Theorem 6.5.8. The proof idea goes as follows. We begin by considering the summand $A_{n,0}$ in the first line of π. We expand $A_{n,0}$ by applying the forward distributivity rule as much as we like (without using the backward distributivity rule, but possibly with applying other rules). In case we expand too much (i.e., exponentially many) summands, we are done. Otherwise, the expansion of $A_{n,0}$ yields a summand Φ which is a simple ancestor of $\psi \times \prod_{k=1}^{m} \Psi_k$, where m is big enough (that is, $m \geq n/2$), and Φ conforms to the conditions in Key Lemma 6.5.5 *excluding* the fact that the initial line of the proof is *not necessarily* Init_n. Since, by assumption, no forward distributivity rule applications are applied on Φ the *backward* distributivity must be applied on it sometime. We can then show that in this case there exists a vertex v in the underlying graph G_π of π that *fully* conforms to the conditions stated in Key Lemma 6.5.5 (more correctly, v fully conforms to the conditions stated in Corollary 6.5.7), which concludes the lower bound proof.

For every $k \in [n]$, we put

$$\Psi_k :\equiv (x_k + b_0)\,.$$

Proposition 6.5.9 *For all $k, i, j \in [n]$, $\Psi_k \notin \mathsf{deriv}(x_i + b_j)$.*

Proof. By the definition of the derivation rules (Definition 6.2.2), no rule can increase the number of plus gates in a formula, except for the forward distributivity rule

$$\frac{Q_1 \times (Q_2 + Q_3)}{(Q_1 \times Q_2) + (Q_1 \times Q_3)} :$$

every plus gate that occurs in (a substitution instance of) Q_1 in the upper-line $Q_1 \times (Q_2 + Q_3)$, appears twice in the lower-line $(Q_1 \times Q_2) + (Q_1 \times Q_3)$. Note that for this increase in the number of plus gates to happen, the upper-line $Q_1 \times (Q_2 + Q_3)$ must contain at least two plus gates. Thus, if Φ' is derived from Φ via an analytic symbolic derivation, and Φ contains only one plus gate, then Φ' contains at most one plus gate. We conclude that for all $i, j \in [n]$, any formula derivable from $(x_i + b_j)$ contains at most one plus gate.

Assume by a way of contradiction that there is a formula $\varphi \in \mathsf{deriv}(x_i + b_j)$, such that there exists $k \in [n]$ for which $(x_k + b_0)$ occurs inside φ. Since φ has only one plus gate,

$\varphi \equiv \phi \times (x_k + b_0)$, for some possibly empty formula ϕ (when ignoring associativity of formulas and the order of multiplication). By the soundness of the proof system, we get that $\phi \times (x_k + b_0)$ and $(x_i + b_j)$ compute the same polynomial. But, by assumption, $b_j \neq b_0$ for all $j \in [n]$, and so for all $k \in [n]$, $(x_i + b_j)$ cannot be factored by $(x_k + b_0)$. □

Fix a regular depth-3 analytic symbolic proof π of Sym_n, and let G_π be the corresponding underlying graph of π. Note that the first line of π is $\mathsf{Sym}_n \equiv A_{n,0} + \Sigma_{j=1}^{n} A_{n,j} - 1$, and so the first level of G_π has a unique vertex labeled with (the product formula) $A_{n,0}$. Let $\mathcal{S}$ be the set of all vertices v in G_π such that $\dot{v}$ is non-atomic (Definition 6.3.2) and every (directed) path that originates in v terminates in the vertex that is labeled with $A_{n,0}$. Let

$$\mathcal{T} \text{ be the subgraph of } G_\pi \text{ induced by the vertices in } \mathcal{S}\,.$$

Proposition 6.5.10 *The graph $\mathcal{T}$ is a binary tree rooted at $A_{n,0}$ (where every vertex is directed from leaves toward the root).*[7]

Proof. This stems directly from the regularity condition on the structure of π. Formally, assume by a way of contradiction that there is a (possibly undirected) cycle C in $\mathcal{T}$, and let u be a vertex in C, where $\mathsf{level}(u)$ is the maximal level in G_π. Since u is in a cycle it has two edges adjacent to two distinct vertices w, v in C. But then the fan-out of u in G_π (considered as a directed graph) is 2, as both v and w must be in a smaller level than the level of u (by assumption on the maximality of $\mathsf{level}(u)$, and since no two vertices in the same level of a proof-graph are connected with an edge). Both v and w have a directed path leading to $A_{n,0}$ in G_π, and so u has two different paths leading to $A_{n,0}$, in contrast to the regularity condition which forbids this case (since, $\mathbb{T}_u(G_\pi)$ is a tree, rooted at u). This shows that $\mathcal{T}$ is a tree. To show that $\mathcal{T}$ is a binary tree, we only need to note that by definition, every vertex in G_π has fan-in at most 2. □

Recall the definition of a subformula in a proof-line being *the consequence* of some derivation rule (Notation after Definition 6.2.1). Also recall that a derivation rule is applied *outside* a subformula ψ if (in the terminology used in Notation after Definition 6.2.1) the vertex v is not in ψ.

Proposition 6.5.11 *No vertex in $\mathcal{T}$ is (labeled with) a consequence of the backward distributivity rule applied outside a $\hat{\Sigma}$ formula.*[8]

Proof. By definition no vertex in G_π is labeled with a plus formula. Hence, all vertices in $\mathcal{T}$ are labeled with (not necessarily proper) $\Pi\hat{\Sigma}$ formulas (since the proof is restricted to $\Sigma\Pi\hat{\Sigma}$ formulas only). The upper-line in the backward distributivity rule is $(Q_1 \times Q_2) + (Q_1 \times Q_3)$.

[7]$\mathcal{T}$ is a tree also when considered as an undirected graph.

[8]Here we identify the vertices in $\mathcal{T}$ with the corresponding vertices in G_π (where the latter correspond, as always, to subformulas occurring in π).

Assume, by a way of contradiction, that there is a vertex v in $\mathcal{T}$, such that v is a consequence of the backward distributivity rule applied on (a substitution instance of) $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ and $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is not a $\hat{\Sigma}$ formula. In case $(Q_1 \times Q_2)$ and $(Q_1 \times Q_3)$ appear in two distinct vertices u, w in G_π, then we get that v has two outgoing edges pointing to u and w. By the definition of $\mathcal{T}$, this means that both u and w have directed paths reaching the root of $\mathcal{T}$, which contradicts the regularity condition.

Otherwise, $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ occurs in the label of some *single* node w. Since no single node is labeled with a plus formula, it must be that w is labeled with a product formula that contains $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ as a subformula. This means that w contains a proper $\Pi\Sigma\Pi$ formula, where the middle $\Sigma\Pi$ formula is not a $\hat{\Sigma}$ formula (since $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is not a $\hat{\Sigma}$ formula). A contradiction to assumption. □

The following proposition exploits our restriction to depth-3 formulas.

Proposition 6.5.12 *Let π be an analytic symbolic proof operating with $\Sigma\Pi\hat{\Sigma}$ formulas. Let v be a node in G_π labeled with a $\Pi\hat{\Sigma}$ formula and let $\ell = \mathsf{level}(v)$. If the forward distributivity rule is applied in $\dot{v}$ (in the ℓth proof-line of π), and the rule is applied outside a $\hat{\Sigma}$ formula and further the consequence of the rule application is not a $\hat{\Sigma}$ formula, then v must have fan-in* 2.

Proof. The idea is that a forward distributivity rule cannot be applied "inside" a product formula (excluding the case when it is applied inside a $\hat{\Sigma}$ formula, or when the consequence of the rule application is a $\hat{\Sigma}$ formula), as this would result in a formula having depth > 3.

Formally, the formula $\dot{v}$ can be written as $L_1 \times \cdots \times L_k$, for some $k \geq 1$, where each L_i is a $\hat{\Sigma}$ formula. If $k = 1$ then the proposition trivially holds, since every application of the forward distributivity rule would be applied inside a $\hat{\Sigma}$ formula, in contrast to the assumption.

Otherwise, $k > 1$. Suppose that an application of the forward distributivity rule on some subformula occurring in $\dot{v}$ was performed outside a $\hat{\Sigma}$ formula. A vertex labeled with a formula on which the forward distributivity is applied either has fan-in 1 or 2 (and not 0). Assume by a way of contradiction that v has fan-in 1 and let u be the vertex that points to v. By the definition of the forward distributivity rule (Definition 6.2.2) $\dot{u}$ must be a product formula (since every vertex is labeled either with a product formula or a single leaf; and a single leaf cannot be the consequence of the forward distributivity rule) that resulted from distributing some subformula $\prod_{i \in I} L_i$ over some $\hat{\Sigma}$ formula $L_r \equiv \Delta_1 + \Delta_2$, for some non-empty $I \subseteq [k]$. This means that u is labeled with

$$\left(\prod_{i \in I} L_i \times \Delta_1 + \prod_{i \in I} L_i \times \Delta_2 \right) \times \prod_{j \in J} L_j \,,$$

for $J = ([k] \setminus I) \setminus \{r\}$, where J is non-empty (as otherwise, u was a plus formula). Therefore, u is a $\Pi\Sigma\Pi$ formula where the middle $\Sigma\Pi$ level consists of formulas that are not $\hat{\Sigma}$ formulas (since by assumption the consequence of the rule application

$\left(\prod_{i\in I} L_i \times \Delta_1 + \prod_{i\in I} L_i \times \Delta_2\right)$ is not a $\hat{\Sigma}$ formula). This contradicts our assumption that all proof-lines are $\Sigma\Pi\hat{\Sigma}$ formulas. We then conclude that v must have two immediate children. □

A *simple path* in a binary tree is (the set of vertices that are included in) any path in the tree that starts in a vertex with a sibling (or else starts in the root) and goes down along the path (further away from the root) until it reaches a vertex that has two sons. Formally, in $\mathcal{T}$, we define simple paths as follows:

Definition 6.5.13 (Simple path) *Let P be the path $(v_1, v_2, \ldots, v_k)$, for $k \geq 1$, in the tree $\mathcal{T}$, beginning with the vertex v_1 and ending in v_k (where v_1 is the vertex closest to the root). If the following three conditions hold then the path P is said to be a* simple path *in $\mathcal{T}$.*

(i) v_1 points to no vertex (in which case v_1 is the root of $\mathcal{T}$), or v_1 points to a vertex v_0 such that v_0 has fan-in 2;

(ii) v_k has fan-in 2;

(iii) all vertices excluding v_k in the path have fan-in at most 1.

Proposition 6.5.14 *Let P be a simple path in $\mathcal{T}$, in which the first (that is, closest to the root) vertex is labeled with a formula from $\mathsf{Cl}\left(\psi \times \prod_{k\in K}(x_k + b_0)\right)$, for some possibly empty formula ψ, and some $K \subseteq [n]$. Then, for every vertex v in P, $\dot{v} \in \mathsf{Cl}\left(\psi \times \prod_{k\in K}(x_k + b_0)\right)$.*

Proof. Assume that P has more than one vertex (as otherwise the statement is trivial). Suppose that all the vertices in the simple path, excluding the first one, are not the consequences of applying one of the two distributivity rules. Thus, the lemma stems directly from the definition of a simple descendant (Definition 6.5.3).

Otherwise, there exists a vertex v in the simple path (different from the first vertex in P) that is a consequence of an application of one of the two distributivity rules.

By Proposition 6.5.11, v is not a consequence of the backward distributivity rule. Then, suppose that v is a consequence of the forward distributivity rule application. In case the forward distributivity rule was applied inside a $\hat{\Sigma}$ formula or the consequence of the forward distributivity rule application is a $\hat{\Sigma}$ formula, then again the lemma stems directly from the definition of a simple descendant. Otherwise, the forward distributivity rule application meets the conditions in Proposition 6.5.12, and so we obtain a vertex in the simple path P (that is, the vertex to which v points) that has fan-in 2, in contrast to the definition of simple paths. □

Proposition 6.5.15 *Let v be a vertex in $\mathcal{T}$ such that $\dot{v}$ is a simple descendant of $\psi \times \prod_{k\in K}(x_k + b_0)$, where ψ is any possibly empty formula and $K \subseteq [n]$, such that $|K| > 1$. Assume that v has two vertices u, w pointing to it. Then, u, w are the consequences of* *applying the forward distributivity rule on $\dot{v}$ (in π), and u, w are labeled with a simple descendant of $\psi' \times \prod_{k\in K'}(x_k+b_0)$ and $\psi'' \times \prod_{k\in K'}(x_k+b_0)$, respectively, where $K' \subseteq K$ and $|K'| \geq |K| - 1$, and ψ' and ψ'' are some two formulas.*

Proof. First note that the only derivation rule that can result in v having two sons is the forward distributivity rule. Thus the forward distributivity rule was applied on $\dot{v}$ in π.

Given an application of the forward distributivity rule, let us call the (substitution instance of) the subformula $(Q_2 + Q_3)$ (in the upper-line of the rule $Q_1 \times (Q_2 + Q_3)$), the *principal sum*. Thus, every application of the forward distributivity rule must break the principal sum into two subformulas Q_2 and Q_3 in the lower-line of the rule $Q_1 \times Q_2 + Q_1 \times Q_3$.

Under the conditions in the statement of the proposition, there are only two cases to consider. The first is that the principal sum of the distributivity rule applied on $\dot{v}$ is a sum $(x_j + b_0)$, for some $j \in K$;[9] the second, is that the principal sum is some subformula in ψ.

In the first case, we obtain that u, w are labeled with a simple descendant of $\psi' \times \prod_{k\in K'}(x_k + b_0)$ and $\psi'' \times \prod_{k\in K'}(x_k + b_0)$, respectively, where $K' = K \setminus \{j\}$ (and so $|K'| = |K| - 1$), and ψ' and ψ'' are some two formulas, which is precisely what we need to show.

In the second case, the principal sum that breaks into two subformulas is a part of ψ, and so both u, w are labeled each with a simple descendant of $\psi' \times \prod_{k\in K}(x_k + b_0)$, and $\psi'' \times \prod_{k\in K}(x_k + b_0)$, respectively, for some two formulas ψ', ψ'', which concludes the proof of the lemma. $\square$

The following proposition is similar to Proposition 6.5.15.

Proposition 6.5.16 *Let w be a vertex in G_π (not necessarily in $\mathcal{T}$) and assume that w has fan-out 2 with two edges pointing to the vertices v and u. Suppose that $\dot{v}$ is a simple descendant of $\psi \times \prod_{k\in K}(x_k + b_0)$, where ψ is some possibly empty formula and $K \subseteq [n]$ such that $|K| > 1$. Then w is the consequence of applying the backward distributivity rule on v and u (in π; in fact the rule was applied on the plus gate that has $\dot{v}$ and $\dot{u}$ as its two sons), and u is labeled with a simple descendant of $\psi' \times \prod_{k\in K'}(x_k + b_0)$, where $K' \subseteq [n]$ and $|K'| \geq |K| - 1$, and ψ' is some possibly empty formula.*

Proof. This is similar to the proof of Proposition 6.5.15. We omit the details. $\square$

Proposition 6.5.17 *If v is a vertex in $\mathcal{T}$, then $\dot{v}$ does not compute the zero polynomial.*

Proof. The root formula of $\mathcal{T}$ is $A_{n,0}$. Without loss of generality, we can assume that r_0 in $A_{n,0}$ is nonzero (there must be at least one nonzero r_i, for $0 \leq i \leq n$, as otherwise Sym_n

[9]Note that one cannot derive from $(x_j + b_0)$ any other formulas (ignoring the order) other than $(1 \times x_j + b_0)$ as any other formula will be a non $\hat{\Sigma}$ formula, which will result in $\dot{v}$ not to be a $\Sigma\Pi\hat{\Sigma}$ formula.

would not compute the zero polynomial). Let P be (the set of vertices in) the path in $\mathcal{T}$ starting in the root and leading to the vertex v. We show by induction on the length of P that every vertex in P is labeled with a formula that can be written as follows (ignoring the order of multiplication and addition):

$$\prod_{k\in K} c_k \times \prod_{j\in J} x_j \times \prod_{i\in I}(d \times x_i + b_0)\,, \tag{6.7}$$

for three possibly empty sets of indices $J, I \subseteq [n]$ and K and where the c_k's are nonzero field elements and d is a possibly empty field element (and so the formula computes a nonzero polynomial).

The base case is immediate (it is the root formula $\mathcal{A}_{n,0}$). For the induction step, we consider all possible rule applications and show that they preserve the induction statement.

The backward distributivity rule is not applicable in P, by definition of $\mathcal{T}$, unless it is applied inside a $\hat{\Sigma}$ formula, which is impossible in this case.

Note that there are no coefficients multiplying the b_0's. This is because by definition of a $\hat{\Sigma}$ formula, such a formula does not contain a product of two or more field elements. Also, notice that every rule, different from the distributivity rules applied on the formula (6.7), keeps the induction statement (this can be verified by straightforward inspection). For the forward distributivity rule, it is easy to see that this rule may only transform a formula of the form (6.7) into two other formulas, each of the form (6.7). $\square$

We now transform the tree $\mathcal{T}$ into a *full binary tree* (that is, a tree in which every vertex has either fan-in 2 or fan-in 0; a full binary tree might not be balanced, though). This is done by contracting all simple paths in $\mathcal{T}$ (that is, contracting all edges pertaining to a simple path). Formally, we perform the following process on $\mathcal{T}$. Let u be a vertex in $\mathcal{T}$ that has fan-in 1 (in $\mathcal{T}$) and denote by w the (single) vertex in $\mathcal{T}$ that points to u. Replace w with u (in other words, the edge from w to u is contracted and the edge(s) going into w now go into u [and u keeps its label]). Continue this process until there are no vertices of fan-in 1 in the graph. We thus obtain a full binary tree with the root being the vertex labeled with $A_{n,0}$. Denote by $\mathcal{T}'$ the graph just constructed.

Notation:

1. We identify the vertices common to both $\mathcal{T}$ and $\mathcal{T}'$.
2. The level of a vertex v in the full binary tree $\mathcal{T}'$ should not be confused with the level $\mathsf{level}(v)$ of the same vertex v in G_π. To avoid confusion we shall explicitly write in what follows *the level of v in $\mathcal{T}'$* when referring to the former measure.

Lemma 6.5.18 *For $0 \le i \le n-1$, let v be a vertex in the ith level of (the full binary tree) $\mathcal{T}'$ (if such a level exists in $\mathcal{T}'$). Then, $\dot{v}$ is a simple descendant of $\psi \times \prod_{k\in K}(x_k + b_0)$, where $K \subseteq [n]$ and $|K| \ge n-i$, and ψ is any possibly empty formula.*

Proof. Note that the root of $\mathcal{T}'$ is labeled with $A_{n,0}$ and that

$$A_{n,0} \equiv r_0 \times \prod_{i\in[n]}(x_i + b_0) \in \mathsf{Cl}\left(\psi \times \prod_{k\in K}(x_k + b_0)\right) , \tag{6.8}$$

for $[n] = K$ and where $\psi = r_0$.

Consider the (not necessarily simple) path P in $\mathcal{T}$ (not in $\mathcal{T}'$) starting from the root and reaching v. By the definition of $\mathcal{T}'$, this path (formally, only the vertices of this path) consists in moving along i consecutive simple paths in $\mathcal{T}$ and then moving one edge further to v (if $i = 0$ then we start from the root of $\mathcal{T}$, and stay there).

By (6.8) and Proposition 6.5.14, all the vertices in the first simple path in P are labeled with elements of $\mathsf{Cl}\left(\psi \times \prod_{k\in K}(x_k + b_0)\right)$.

Starting in the last vertex of the first simple path we can move along P (further away from the root) to its son w. By definition the last vertex of every simple path has two sons (and so w has a sibling in $\mathcal{T}$). Hence, we can apply Proposition 6.5.15 to conclude that $\dot{w} \in \mathsf{Cl}\left(\psi' \times \prod_{k\in K}(x_k + b_0)\right)$, for some $|K| \subseteq [n]$ such that $|K| \geq n - 1$, and where ψ' is some possibly empty formula.

Using the same reasoning, after moving along P through i consecutive simple paths, and then moving one edge down to v, we conclude that $\dot{v} \in \mathsf{Cl}\left(\psi' \times \prod_{k\in K}(x_k + b_0)\right)$, where $K \subseteq [n]$ such that $|K| \geq n - i$, and ψ' is some possibly empty formula. □

Concluding the proof of Theorem 6.5.8. Let v be a *leaf* in $\mathcal{T}'$ having the minimal level ℓ *in the full binary tree* $\mathcal{T}'$.

Case 1: $\ell \geq n/2$. In this case $\mathcal{T}'$ is a full binary tree where all leaves are of level at least $n/2$. This means that the number of vertices in $\mathcal{T}'$ is at least as the number of vertices of a *complete* binary tree with $n/2$ levels (a complete binary tree is a full and balanced binary tree). Thus, the number of vertices in $\mathcal{T}'$ is at least $2^{n/2} - 1$. Since every vertex in $\mathcal{T}'$ appears in $\mathcal{T}$ and thus also appears in G_π, we get that $|G_\pi| = 2^{\Omega(n)}$.

Case 2: $\ell < n/2$. By Proposition 6.5.18:

$$\dot{v} \in \mathsf{Cl}\left(\psi \times \prod_{k\in K}(x_k + b_0)\right) , \tag{6.9}$$

where $K \subseteq [n]$ such that $|K| \geq n - \ell > n/2$ (and ψ is some possibly empty formula).

From now on, we will consider v as a vertex in G_π. By the definition of $\mathcal{T}$, there are only two cases that account for v being a leaf in $\mathcal{T}$: (i) the vertex v is of fan-in 0 in G_π; or (ii) there is a (unique) vertex w in G_π that has an out-going edge pointing to v and further there is a directed path in G_π starting from w and terminating in (the first level of G_π) in a vertex different from the root of $\mathcal{T}$ (i.e., different from the vertex labeled with $A_{n,0}$).

In case (i), v must be labeled with the formula 0 (this can be checked by inspection of the derivation rules [Definition 6.2.2]). But by Proposition 6.5.17, $\dot{v}$ does not compute the zero polynomial, and so we arrive at a contradiction.

In case (ii), the vertex w has fan-out 2 (since every directed path in G_π starting in v terminates in the root of $\mathcal{T}$ by definition of $\mathcal{T}$). Thus, w has an out-going edge that goes into v and another out-going edge that goes into a vertex we denote by u. Now, by (6.9) and Proposition 6.5.16 we conclude that u is labeled with a simple descendant of $\psi \times \prod_{k\in K}(x_k + b_0)$, where $K \subseteq [n]$ such that $|K| \geq n - \ell - 1 > n/2 - 1$ (and ψ is some possibly empty formula).

By the regularity condition we have that the trees $\mathbb{T}_v(G_\pi)$ and $\mathbb{T}_u(G_\pi)$ have no common vertices. This in turn means that the conditions of Corollary 6.5.7 hold for the vertex u (since, by regularity, u does not have a directed path that reaches $A_{n,0}$ in the initial line in π, and so every directed path beginning at u must terminate in the initial line in a summand pertaining to Init_n) which implies that $|\mathbb{T}_u(G_\pi)| > 2^{n/2-1}$, and so we conclude that $|G_\pi| = 2^{\Omega(n)}$.

6.5.4 Proof of Technical Lemma 6.5.6

Here we prove the main technical lemma. For the sake of convenience, we repeat it fully here:

Lemma 6.5.6 (Technical lemma) *Let π be a regular analytic symbolic proof operating with $\Sigma\Pi\hat{\Sigma}$ formulas, and with the initial formula in π being Init_n, for some positive $n \in \mathbb{N}$. Let G_π be the corresponding graph of π. Assume that v is a vertex in G_π labeled with Φ, which is a simple ancestor of*

$$\psi \times \prod_{k=1}^{m} \Psi_k\,,$$

where $m > 1$ and ψ is any possibly empty formula, and for every $k \in [m]$, $i \in [n]$ and $j \in [n]$, Ψ_k is a proper $\hat{\Sigma}$ formula, such that $\Psi_k \notin \mathsf{deriv}(x_i + b_j)$. Then, there exists a vertex y in $\mathbb{T}_v(G_\pi)$, such that there is a path from v to y in $\mathbb{T}_v(G_\pi)$, and y has two outgoing edges to two (distinct) vertices u, w, so that: u and w are labeled with two simple ancestors of the following product formulas

$$\psi_0 \times \prod_{k=1}^{m-1} \Psi'_k \qquad \textit{and} \qquad \psi_1 \times \prod_{k=1}^{m-1} \Psi''_k\,, \tag{6.10}$$

respectively, where ψ_0, ψ_1 are some possibly empty formulas, and for all $k \in [m-1]$, $i \in [n]$ and $j \in [n]$, Ψ'_k, Ψ''_k are some proper $\hat{\Sigma}$ formulas such that $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$ and $\Psi''_k \notin \mathsf{deriv}(x_i + b_j)$.

Denote by $\mathscr{F}$ the set of simple ancestors of all formulas $\prod_{k=1}^{m} \Psi_k$ where each Ψ_k (from the statement of the lemma) is substituted by some Ψ'_k, such that Ψ_k is a derivable *sub*formula from Ψ'_k. Formally, we have:

$$\mathscr{F} := \left\{ \mathsf{Cl}^{-}\left(\prod_{k=1}^{m} \Psi'_k\right) \;\middle|\; \forall k \in [m],\ \Psi_k \in \mathsf{deriv}(\Psi'_k) \right\}.$$

By the definitions of a simple closure (Definition 6.5.3) and of derivable subformulas (Definition 6.5.2), we have the following simple properties:

Fact 6.5.19 *1.* $\mathsf{deriv}(\cdot)$ *is transitive: if* $\phi_0 \in \mathsf{derive}(\phi_1)$ *and* $\phi_1 \in \mathsf{derive}(\phi_2)$*, then* $\phi_0 \in \mathsf{derive}(\phi_2)$.

2. *For all* $\phi \in \mathsf{Cl}^-(\Psi)$*,* $\Psi \in \mathsf{deriv}(\phi)$*. (On the other hand,* $\Psi \in \mathsf{deriv}(\phi)$ *does not necessarily imply* $\phi \in \mathsf{Cl}^-(\Psi)$ *(see Definition 6.5.2).)*

3. *From the above two facts: If* $\Psi_k \in \mathsf{deriv}(\Psi'_k)$*, then for all* $\phi \in \mathsf{Cl}^-(\Psi'_k)$ *it holds that* $\Psi_k \in \mathsf{deriv}(\phi)$*.*

Proposition 6.5.20 *Every formula in* $\mathscr{F}$ *is a simple ancestor of* $\prod_{k=1}^{m} \Psi'_k$*, where, for every* $k \in [m]$*,* $i \in [n]$ *and* $j \in [n]$*,* $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$*, and* Ψ'_k *contains at least one plus gate.*[10]

Proof. Assume by a way of contradiction that $\Psi'_k \in \mathsf{deriv}(x_i + b_j)$, for some $k \in [m]$, $i \in [n]$ and $j \in [n]$. By $\Psi_k \in \mathsf{deriv}(\Psi'_k)$, and by the transitivity of $\mathsf{deriv}(\cdot)$, we have that $\Psi_k \in \mathsf{deriv}(x_i + b_j)$, which contradicts the assumption on the Ψ_k's (in Lemma 6.5.6).

Further, since $\Psi_k \in \mathsf{deriv}(\Psi'_k)$, for every $k \in [m]$, and since Ψ_k is a proper $\hat{\Sigma}$ formula, every Ψ'_k must contain at least one plus gate (this can be verified by inspecting the definition of $\mathsf{deriv}(\cdot)$ and the derivation rules [Definition 6.2.2]). $\square$

Proposition 6.5.21 *No member of* $\mathscr{F}$ *occurs as a subformula in the initial proof-line.*

Proof. Every formula in $\mathscr{F}$ must contain at least $m > 1$ products of ψ_k's, such that $k \in [m]$ and $\psi_k \in \mathsf{Cl}^-(\Psi'_k)$ and $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$, for all $i \in [n]$ and $j \in [n]$. By Item 3 in Fact 6.5.19 and by $\Psi_k \in \mathsf{deriv}(\Psi'_k)$ we get that $\Psi_k \in \mathsf{deriv}(\psi_k)$. By assumption (Lemma 6.5.6), $\Psi_k \notin \mathsf{deriv}(x_i + b_j)$, for all $i \in [n]$ and $j \in [n]$, and so $\psi_k \notin \mathsf{deriv}(x_i + b_j)$ (for all $i \in [n]$ and $j \in [n]$).

On the other hand, no formula in $\mathscr{F}$ contains products of (non empty) ψ_k's for which $\psi_k \notin \mathsf{deriv}(x_i + b_j)$, for all $i \in [n]$ and $j \in [n]$. This concludes the proof. $\square$

Proposition 6.5.22 (Critical transition) *There is a directed path (with* 0 *or more edges) from* v *to some vertex* y *in* $\mathbb{T}_v(G_\pi)$*, where* $\mathsf{level}(v) \geq \mathsf{level}(y) > 1$*, such that:* $\dot{y}$ *contains a subformula which is an element of* $\mathscr{F}$*, while the preceding level* $\mathsf{level}(y) - 1$ *corresponds to a proof-line that does not contain a subformula which is an element of* $\mathscr{F}$*.*

[10]When considering only $\Sigma\Pi\hat{\Sigma}$ formulas, the condition that Ψ'_k contains at least one plus gate may be replaced by the condition that Ψ'_k is a proper $\hat{\Sigma}$ formula (note that in the definition of $\mathscr{F}$ we have not restricted the depth of formulas).

(Note that $m > 1$ and so Φ and all formulas in $\mathscr{F}$ are product formulas, which means that every formula in $\mathscr{F}$ may occur [as a complete formula] only in a label of at most one vertex in each level.[11])

Proof. Assume by a way of a contradiction that there is no such y. Note that $\dot{v}$ contains a subformula from $\mathscr{F}$, since clearly $\prod_{k=1}^{m} \Psi_k \in \mathscr{F}$. It is evident (by Definition 6.3.1) that every vertex in a proof-graph has a path originating in that vertex and terminating in the first level of the graph. Therefore, since there is no y that meets the conditions stated in the claim, every vertex in G_π, on the path from v to a vertex in the first level, must contain a subformula from $\mathscr{F}$. Thus, the initial proof line contains a subformula from $\mathscr{F}$, which contradicts Proposition 6.5.21. □

To conclude for now, let y be the vertex whose existence is guaranteed by Proposition 6.5.22, put $\ell = \mathsf{level}(y)$ and let $\mathcal{A}\{\Theta\}$ denote the formula that corresponds to the (whole) level ℓ, where

$$\Theta :\equiv \dot{y}$$

(note that since Θ is the label of a vertex, Θ is in fact just a summand in $\mathcal{A}$). Then we can write

$$\frac{\mathcal{A}\{\Theta'\}_t}{\mathcal{A}\{\Theta\}_t}\ (\star) \tag{6.11}$$

to denote the transformation made from level (i.e., proof-line) $\ell - 1$ to level (i.e., proof-line) ℓ, obtained by some derivation rule $(\star)$, where t is a node in $\mathcal{A}$ and:

$$\text{there exists a subformula } \theta \text{ in } \Theta \text{ such that } \theta \in \mathscr{F} \tag{6.12}$$

$$\text{there is no subformula } \theta \text{ in } \mathcal{A}\{\Theta'\}_t \text{ such that } \theta \in \mathscr{F}. \tag{6.13}$$

Proposition 6.5.23 *Both the formula Θ and the formula θ are proper $\Pi\hat{\Sigma}$ formulas (Definition 6.5.4).*

Proof. By definition, Ψ_k is a *proper* $\hat{\Sigma}$ formula, for every $k \in [m]$. Since $\Psi_k \in \mathsf{derive}(\Psi'_k)$ for all $k \in [m]$, Ψ'_k must contain some plus formula, for all $k \in [m]$ (this can be verified by inspection of the derivation rules [Definition 6.2.2]). Hence, since $m > 1$, $\theta \in \mathscr{F}$ must contain a product of at least two formulas, each of which contains a plus formula. Therefore, also the formula Θ must contain a product of at least two formulas, each of which contains a plus formula. Since Θ is a label of some vertex, it must be that Θ in this case is a product formula. And since we work with $\Sigma\Pi\hat{\Sigma}$ formulas, Θ must be a proper $\Pi\hat{\Sigma}$ formula. Further, since θ occurs in Θ (and θ contains a product of two proper $\hat{\Sigma}$ formulas), θ is also a proper $\Pi\hat{\Sigma}$ formula. □

In order to prove Lemma 6.5.6 we will demonstrate that y above is the vertex stated in the statement of this lemma. Specifically, we will consider all possible rules $(\star)$ that can

[11] In other words, there are no two vertices s, t, at the same level and a formula $\Theta \in \mathscr{F}$, such that one subformula of Θ occurs in $\dot{s}$ and a different subformula of Θ occurs in $\dot{t}$.

be applied in (6.11) above, and conclude that there must be two outgoing edges from y into two vertices that meet the requirements of Lemma 6.5.6.

Notation: Assume, for example, that the rule $(\star)$ applied in (6.11) is $\dfrac{Q_1 + 0}{Q_1}$, and let φ be some formula. Then we shall say, for instance, that *φ occurs in the upper-line* $Q_1 + 0$, to mean that φ occurs in some *substitution instance* Δ of $Q_1 + 0$ (and that clearly Δ occurs in $\mathcal{A}\{\Theta'\}_t$ from (6.11)). In other words, when referring to occurrences in upper and lower lines of rules that are formulated with formulas in the variables Q_1, Q_2, Q_3, we are formally referring to *substitution instances* of these variables.

Definition 6.5.24 (Closure of $\mathscr{F}$ under derivation rules) *We say that $\mathscr{F}$ is* closed under some derivation rule $(\star)$ (or under certain instances of the derivation rules)*, if whenever a formula Δ is transformed via $(\star)$ (or via a certain instance of the derivation rule) into Δ', and Δ' contains a subformula in $\mathscr{F}$, then Δ also contains a subformula in $\mathscr{F}$.* [12]

Case 1: The rule $(\star)$ is one of the following: commutativity, associativity, unit element rules, zero element rules or scalar rules; or the rule $(\star)$ is a distributivity rule applied in a $\hat{\Sigma}$ formula or the consequence of applying $(\star)$ is a $\hat{\Sigma}$ formula (the last two options correspond to the transformations made in the last two clauses in Definition 6.5.3). By the definition of $\mathscr{F}$ (and by the definition of a simple closure [Definition 6.5.3]), $\mathscr{F}$ is closed under these rules (Definition 6.5.24). Thus, proof-line $\ell - 1$ contains a formula which is an element in $\mathscr{F}$, and we arrive at a contradiction with (6.13).

Case 2: The rule $(\star)$ is forward distributivity applied differently from that in Case 1 (that is, it is not applied inside a $\hat{\Sigma}$ formula and its consequence is not a $\hat{\Sigma}$ formula):

$$\frac{Q_1 \times (Q_2 + Q_3)}{(Q_1 \times Q_2) + (Q_1 \times Q_3)}.$$

We consider the possible occurrences of the lower line $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ in $\mathcal{A}\{\Theta\}_t$, and conclude that this case does not hold.

(i) There is no subformula of Θ that occurs in the lower line $(Q_1 \times Q_2) + (Q_1 \times Q_3)$.

Thus, proof-line $\ell - 1$ contains Θ as a subformula, in contrast to (6.13).

(ii) (A substitution instance of) $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is a *proper* subformula of Θ.

By assumption (made in Case 2) $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is not a $\hat{\Sigma}$ formula. Note that $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is a proper $\Sigma\Pi$ formula when considered as a formula in the propositional variables Q_1, Q_2, Q_3 (and not necessarily as a substitution instance of these variables). Since $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is not a $\hat{\Sigma}$ formula and since Θ is a proper $\Pi\hat{\Sigma}$ formula (Proposition 6.5.23), we arrive at a contradiction.

[12]Note that here we demand that $\mathscr{F}$ is closed under a derivation rule, if it is closed when the rule is applied "backward" on a formula in $\mathscr{F}$.

(iii) The formula Θ is (a substitution instance of the whole formula) $(Q_1 \times Q_2) + (Q_1 \times Q_3)$.

This is contradictory to Θ being a product formula (Proposition 6.5.23).

(iv) The formula Θ is (a substitution instance of) $(Q_1 \times Q_2)$.

Suppose that Q_2 is not a $\hat{\Sigma}$ formula. Then it ought to be a proper $\Pi\hat{\Sigma}$ (by Proposition 6.5.23). Thus, $Q_1 \times (Q_2 + Q_3)$ (from the upper-line) is a proper $\Pi\Sigma\Pi\hat{\Sigma}$ formula that appears in proof-line $\ell - 1$ in π. This contradicts our assumption that all proof-lines are $\Sigma\Pi\hat{\Sigma}$ formulas.

Therefore, Q_2 is a $\hat{\Sigma}$ formula. Assume that Q_2 is $\hat{\Sigma}$ formula denoted Δ that occurs in $\theta \in \mathscr{F}$ (see (6.12)). Then, the upper line $Q_1 \times (Q_2 + Q_3)$ is just Θ with Q_3 added to one of the $\hat{\Sigma}$ products in it. Note that if $F \in \mathscr{F}$ and F' is the result of adding some formula to one of the $\hat{\Sigma}$ products occurring inside F, then $F' \in \mathscr{F}$; this is because for every two formulas Δ_1, Δ_2, $\Psi_k \in \mathsf{deriv}(\Delta_1)$ implies $\Psi_k \in \mathsf{deriv}(\Delta_1 + \Delta_2)$ (by definition of $\mathsf{deriv}(\cdot)$).[13] Thus, the upper line $Q_1 \times (Q_2 + Q_3)$ still contains an element of $\mathscr{F}$ in contrast with (6.13).

(v) The formula Θ is $(Q_1 \times Q_3)$. This is analogous to the previous sub-case (iv).

(vi) The formula Θ is a substitution instance of one of Q_1 or Q_2 or Q_3.

Thus, Θ occurs also in the upper-line (and hence in line $\ell - 1$ of the proof), which contradicts (6.13).

Case 3: The rule $(\star)$ is the backward distributivity

$$\frac{(Q_1 \times Q_2) + (Q_1 \times Q_3)}{Q_1 \times (Q_2 + Q_3)},$$

applied differently from that in Case 1 (that is, it is not applied inside a $\hat{\Sigma}$ formula and its consequence is not a $\hat{\Sigma}$ formula[14]). Thus, $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is not a $\hat{\Sigma}$ formula.

We shall consider the possible occurrences of the lower-line $Q_1 \times (Q_2 + Q_3)$ in $\mathcal{A}\{\Theta\}_t$, and conclude that the upper-line $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ constitutes in $\mathbb{T}_v(G_\pi)$ the two vertices u, w to which y points, and that u, w meet the conditions stated in the lemma (Lemma 6.5.6).

(i) There is no subformula of θ that occurs in the lower-line $Q_1 \times (Q_2 + Q_3)$.

Thus, proof-line $\ell - 1$ contains θ as a subformula, in contrast to (6.13).

[13] Here we use the fact that $\mathsf{deriv}(\cdot)$ is defined as the derivable *sub*formulas, and not just derivable formulas.

[14] In the case of the backward distributivity rule, these two options are the same, since the latter transformation is also applied inside a $\hat{\Sigma}$ formula.

(ii) There is a only a *proper* subformula of θ that occurs in $Q_1 \times (Q_2 + Q_3)$ (that is, θ does not occur fully in $Q_1 \times (Q_2 + Q_3)$).

Recall that θ is a product formula (Proposition 6.5.23), and so we can write θ as $\Pi_{i \in I}\theta_i$ for some set of formulas θ_i, $i \in I$. The only possibility in the current case is that there is a partition of I into two nonempty subsets of indices $I = I_0 \uplus I_1$, so that θ can be partitioned into two products; one is the product of all θ_i, $i \in I_0$, and the other is the product of all θ_i, $i \in I_1$ (we ignore the order in which the θ_i's occur in θ), and such that: either Q_1 or $(Q_2 + Q_3)$ or $Q_1 \times (Q_2 + Q_3)$ is the formula $\prod_{i \in I_0} \theta_i$ and the product of all θ_i, $i \in I_1$, *does not occur in the lower-line* $Q_1 \times (Q_2 + Q_3)$. Since $\theta \equiv \Pi_{i \in I}\theta_i$, it must be that $\prod_{i \in I_1} \theta_i$ *multiplies* $Q_1 \times (Q_2 + Q_3)$ inside $\mathcal{A}\{\Theta\}_t$. This means that in $\mathcal{A}\{\Theta'\}_t$, the formula $\prod_{i \in I_1} \theta_i$ multiplies $(Q_1 \times Q_2) + (Q_1 \times Q_3)$. Since $(Q_1 \times Q_2) + (Q_1 \times Q_3)$ is not a $\hat{\Sigma}$ formula, $\dot{y}$ contains a proper $\Pi\Sigma\Pi$ formula, where the middle $\Sigma\Pi$ level does not consist of only $\hat{\Sigma}$ formula, which contradicts our assumption that all proof-lines are $\Sigma\Pi\hat{\Sigma}$ formulas.

(iii) The formula θ occurs (fully) in $Q_1 \times (Q_2 + Q_3)$.

(1) θ occurs in one of Q_1, Q_2, Q_3. Thus, θ occurs also in the upper-line (and hence in line $\ell - 1$ of the proof), which contradicts (6.13).[15]

(2) θ occurs in $(Q_2 + Q_3)$ and item (iii1) above does not hold. This is impossible since θ is a product formula (Proposition 6.5.23).

(3) θ occurs in $Q_1 \times (Q_2 + Q_3)$, and the above two items (iii1) and (iii2) do not hold. Assume that there exists a proper subformula τ of θ such that $\tau \in \mathscr{F}$ and τ occurs fully in Q_1. Hence, the upper-line also contains τ and we arrive at a contradiction with (6.13).

Otherwise, we are left only with the following case to consider. Write θ as $\Pi_{i \in I}\theta_i$, for some $\hat{\Sigma}$ formulas θ_i, $i \in I$ (since θ is a proper $\Pi\hat{\Sigma}$ formula [Proposition 6.5.23], it is possible to write θ in this way). It must be that there exists a $j \in I$ such that $\theta_j \equiv \Delta_0 + \Delta_1$, where Δ_0, Δ_1 are Q_2, Q_3, respectively; and Q_1 is

$$\left(\psi \times \prod_{i \in I \setminus \{j\}} \theta_i\right),$$

where ψ is some possibly empty formula. Therefore, $(Q_1 \times Q_2)$ and $(Q_1 \times Q_3)$ from the upper-line are:

$$\left(\psi \times \prod_{i \in I \setminus \{j\}} \theta_i\right) \times \Delta_0 \quad \text{and} \quad \left(\psi \times \prod_{i \in I \setminus \{j\}} \theta_i\right) \times \Delta_1,$$

respectively. Thus, we only need to show that these are precisely the two vertices u, w as stated in the lemma (Equation (6.10)). The proof of this is straightforward and we show it formally for the sake of completeness:

[15]This subcase can be shown impossible to meet also due to the depth-3 restriction.

Claim: The formula $\left(\psi \times \prod_{i\in I\setminus\{j\}} \theta_i\right) \times \Delta_0$ is a simple ancestor of the product formula $\psi_0 \times \prod_{k\in K} \Psi'_k$, where ψ_0 is any, possibly empty, formula and $K \subseteq [m]$ such that $|K| = m-1$ and for every $k \in K$, $i \in [n]$ and $j \in [n]$, Ψ'_k is a proper $\hat{\Sigma}$ formulas such that $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$. (The proof of the right hand side formula $\left(\psi \times \prod_{k\in I\setminus\{j\}} \theta_i\right) \times \Delta_1$ is similar.)

Proof of claim: We know that $\Pi_{i\in I}\theta_i \in \mathscr{F}$, and so by Proposition 6.5.20 $\Pi_{i\in I}\theta_i$ is a simple ancestor of $\psi_0 \times \prod_{k=1}^{m} \Psi'_k$, where ψ_0 is some possibly empty formula and for every $k \in [m]$, $i \in [n]$ and $j \in [n]$, $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$, where Ψ'_k contains at least one plus gate. Since each Ψ'_k contains at least one plus gate and every proof-line is a $\Sigma\Pi\hat{\Sigma}$ formula then in fact each Ψ'_k is a proper $\hat{\Sigma}$ formula.

Therefore, $\left(\psi \times \prod_{k\in I\setminus\{j\}} \theta_i\right) \times \Delta_0$ constitutes a simple ancestor of $\psi_0 \times \prod_{k\in K} \Psi'_k$, where ψ_0 is some possibly empty formula and $k \in K$ where $K \subseteq [m]$ and $|K| = m-1$, and for all $i \in [n]$ and $j \in [n]$ and $k \in K$, Ψ'_k is a proper $\hat{\Sigma}$ formulas such that $\Psi'_k \notin \mathsf{deriv}(x_i + b_j)$. $\blacksquare_{\text{Claim}}$

This concludes the proof of Lemma 6.5.6.

6.6 Chapter Summary

In this chapter we studied the complexity of symbolic proofs that establish polynomial identities. This study is motivated by connections with semantic algebraic propositional proof system, and by connections to the problem of polynomial identity testing. We introduced an analytic fragment of symbolic proofs of polynomial identities, which is analogous to some extent with the notion of analytic proofs in propositional logic. We then proved an exponential lower bound on the size of analytic symbolic proofs operating with depth-3 arithmetic formulas under a certain regularity condition on the structure of proofs. The hard instances we used were based on small formulas for the symmetric polynomials (over large enough fields) that we already met in previous chapters.

Chapter 7

Alternative Models of Refutations: Propositional Proofs under a Promise

In this chapter we introduce and study sound and complete propositional proof systems for the set of unsatisfiable CNF formulas under the promise that every CNF formula is either unsatisfiable or has many satisfying assignments.

7.1 Preliminaries and Notations

Let A, B be two propositional formulas. We write $A \equiv B$ as an abbreviation for $(A \to B) \wedge (B \to A)$. The notation $A \not\equiv B$ abbreviates $\neg(A \equiv B)$. We say that A *semantically implies* B, denoted by $A \models B$, iff every satisfying assignment to A also satisfies B.

A kCNF formula is a CNF with all clauses containing k literals each. The *width* of a clause D is the number of literals in it, denoted $|D|$. The *size* of a CNF formula K is the total number of clauses in it, denoted $|K|$. The *width of a CNF formula* K is the maximum width of a clause in K. We denote by $K' \subseteq K$ that K' is a sub-collection of clauses from K.

Recall the definition of the resolution refutation system (Definition 2.2.1 in the preliminaries Chapter 2). For an unsatisfiable CNF formula K the *resolution refutation size of* K is the minimal size of a resolution refutation of K and is denoted $S(K)$. Similarly, the *resolution refutation width of* K is the minimal width of a resolution refutation of K and is denoted $w(K)$. If the family of CNF formulas $\{K_n \mid n \in \mathbb{N}\}$ has polynomial-size resolution refutations we say that resolution can *efficiently certify* the unsatisfiability of K_n. Similarly, if the family of clauses $\{D_n \mid n \in \mathbb{N}\}$ has polynomial-size resolution proofs from K_n we say that D_n *is efficiently provable from* K_n.

7.1.1 Size-Width Tradeoffs

We now recall the general approach for proving size lower bounds on resolution refutations developed by Ben-Sasson and Wigderson (2001). The basic idea is that a lower bound on the resolution refutation width of a CNF formula K implies a lower bound on the resolution refutation size of K:

Theorem 7.1.1 (Ben-Sasson and Wigderson (2001)) *Let K be a CNF formula of width r, then*

$$S(K) = \exp\left(\Omega\left(\frac{(w(K)-r)^2}{n}\right)\right).$$

7.1.2 Boolean Circuit Encoding

The promise axioms we introduce use Boolean circuits to define the set of assignments to be discarded (see Section 7.2). Therefore, as resolution operates only with clauses, we need to encode Boolean circuits as collections of clauses (CNF formulas). We assume that all Boolean circuits use only three gates: $\vee, \wedge, \neg$ (though this is not necessary) where $\vee$ (denoting OR) and $\wedge$ (denoting AND) have fan-in 2 and $\neg$ (denoting NOT) has fan-in 1. Let C be a Boolean circuit with m input bits and n output bits. Let $\overline{W} = \{w_1, \dots, w_m\}$ be the m input variables of C and let X denote the n variables $\{x_1, \ldots, x_n\}$. We consider the n output bits of C as the outputs of n distinct circuits $C_1(\overline{W}), \ldots, C_n(\overline{W})$ in the $\overline{W}$ variables, and we write $C(\overline{W}) \equiv X$ to mean that X equals the output of $C(\overline{W})$ (that is, $C_1(\overline{W}) \equiv x_1 \wedge \cdots \wedge C_n(\overline{W}) \equiv x_n$). This notation can be extended in a similar manner to $C(\overline{W_1}) \equiv C'(\overline{W_2})$ and $C(\overline{W_1}) \not\equiv C'(\overline{W_2})$.

There exists a CNF formula F (in both the $\overline{W}$ variables and new extension variables) that *encodes the circuit* C. This means that there are n new extension variables (among other extension variables) $y_1, \ldots, y_n$ in F such that for all assignments $\overline{a}$: $F(\overline{a}) = 1$ iff $C(w_1(\overline{a}), \ldots, w_m(\overline{a})) = y_1(\overline{a}) \circ \cdots \circ y_n(\overline{a})$, where we denote by $w_i(\overline{a})$ the truth value of w_i under the assignment $\overline{a}$ and by $\circ$ the concatenation of Boolean bits. In other words, F expresses the fact that $y_1, \ldots, y_n$ are the output bits of C. If C is of size s (that is, the number of Boolean gates in C is s), then the size of F is $O(s \cdot \log(s))$. Therefore, if C is of size polynomial in n then F is also of polynomial-size in n. We denote by $\|C(\overline{W})\|$ the CNF formula F that encodes $C(\overline{W})$.

For most purposes, we will not need an explicit description of how the encoding of Boolean circuits as CNF formulas is done through $\|C(\overline{W})\|$. Nevertheless, in Section 7.3 we need to ensure that resolution can efficiently prove several basic facts about the encoded circuits. For this reason, and for the sake of concreteness of the promise axioms (Definitions 7.2.3 and 7.2.4) we provide the precise definition of the encoding in the appendix (Section 7.5.1), in addition to proving some of the encoding's basic (proof theoretical) properties. The interested reader can look at the appendix for any missing details, but anyone willing to accept the existence of an efficient CNF encoding of Boolean circuits that is also intensional for resolution (in the sense that resolution can efficiently prove basic properties of the encoded circuits) can skip Section 7.5.1 without risk.

7.2 Promise Proof Systems

In this section we define precisely the model of refutations under a promise. As discussed in the introduction chapter (Section 1.2.4), we work with the resolution refutation system as our underlying system and augment it with a new set of axioms that we call the *promise axioms*. We call this proof system *promise resolution*. The promise axioms are meant to express the fact that we can discard a certain number of truth assignments from the space of all truth assignments and still be able to certify (due to the promise) whether the input CNF is unsatisfiable or not. Each promise resolution refutation can use at most one promise axiom.

From now on, throughout this chapter, we shall assume that the underlying variables of the CNF formulas that are meant to be refuted are taken from the set $X := \{x_1, \ldots, x_n\}$. The X variables are called the *original variables*. Any other variable that appears in a (promise resolution) refutation is called an *extension variable*.

Definition 7.2.1 (CNF formulas under a promise) *Let Λ be a fixed function in n (the number of X variables) such that $0 \leq \Lambda(n) \leq 2^n$. The function Λ is called the* promise. *The set of* CNF formulas under the promise Λ *consists of all CNF formulas in the X*

variables that are either unsatisfiable or have more then $\Lambda(n)$ satisfying assignments (for $n = |X|$).

The refutation systems we build are sound and complete for the set of CNF formulas under a (given) promise. That is, every unsatisfiable CNF formula has a refutation in the system (this corresponds to completeness), while no CNF having n variables and more than $\Lambda(n)$ satisfying assignments has a refutation in it (this corresponds to soundness under the promise). Soundness (under the promise) is achieved by requiring that resolution should *prove the fact that we discard the right number of assignments* (see Section 7.2.1 for details).

Recall the notion of assignment discarding discussed in the introduction chapter (Section 1.2.4). Formally we have:

Definition 7.2.2 (Assignment discarding) *Let A be a CNF in the X variables that can contain (but does not necessarily contain) extension variables (that is, variables not from X). We say that an assignment to the X variables $\overline{a}$ is* discarded *by A if there is no extension of $\overline{a}$ (to the extension variables in A) that satisfies A.*

7.2.1 Promise Axioms

7.2.1.1 Big Promise

We first concentrate on a promise of a *constant fraction of assignments* (for a smaller promise the axiom is similar; see below).

Let the promise (see Definition 7.2.1) be $\Lambda = \varepsilon \cdot 2^n$, for a constant $0 < \varepsilon < 1$ (we fix this Λ throughout this subsection), and let $r = \lceil \log(1/\varepsilon) \rceil$ and $t = 2^r - 1$. Let C be a sequence of Boolean circuits $C := (C^{(1)}, \ldots, C^{(t)})$. Assume that each $C^{(i)}$ has $n - r$ input bits and n output bits and computes the Boolean map $f_i : \{0,1\}^{n-r} \to \{0,1\}^n$. Assume further that the f_i's are all injective maps and that the images of all these maps are pairwise disjoint. Denote by $\mathsf{Im}(f_i)$ the image of the map f_i. For simplicity, we call the union $\cup_{i=1}^{t} \mathsf{Im}(f_i)$ *the image of C* and denote it by $\mathsf{Im}(C)$. By the definition of r, we have $2^{n-r} \le \varepsilon \cdot 2^n$, and by the injectivity and pairwise disjointness of the images of the f_i's we have:

$$|\mathsf{Im}(C)| = t \cdot 2^{n-r} = (2^r - 1) \cdot 2^{n-r} = 2^n - 2^{n-r} \ge 2^n - \Lambda\,. \tag{7.1}$$

Therefore, *we can treat $\mathsf{Im}(C)$ as the set of all possible truth assignments for the original variables X, without losing soundness*: If K is unsatisfiable then there is no assignment in $\mathsf{Im}(C)$ that satisfies K; and if K is satisfiable then according to the promise it has more than Λ satisfying assignments, which means that there is at least one assignment in $\mathsf{Im}(C)$ that satisfies K. This idea is formulated as a propositional formula as follows:

Definition 7.2.3 (Promise Axiom for $\Lambda = \varepsilon \cdot 2^n$) *Let the promise be $\Lambda = \varepsilon \cdot 2^n$, for a constant $0 < \varepsilon < 1$, and let $r = \lceil \log(1/\varepsilon) \rceil$ and $t = 2^r - 1$. Let C be a sequence of*

Boolean circuits $C := (C^{(1)}, \ldots, C^{(t)})$. Assume that each $C^{(i)}$ has $n-r$ input bits and n output bits and let $\overline{W_1}$ and $\overline{W_2}$ be two disjoint sets of $n-r$ extension variables each. The promise axiom $\mathrm{PRM}_{C,\Lambda}$ *is the CNF encoding (via the encoding defined in Section 7.5.1) of the following Boolean formula:*

$$\left(\bigwedge_{i=1}^{t} \left(C^{(i)}(\overline{W_1}) \equiv C^{(i)}(\overline{W_2}) \to \overline{W_1} \equiv \overline{W_2} \right) \wedge \bigwedge_{1 \le i < j \le t} C^{(i)}(\overline{W_1}) \not\equiv C^{(j)}(\overline{W_2}) \right) \longrightarrow \bigvee_{i=1}^{t} C^{(i)}(\overline{W_1}) \equiv X.$$

The promise axiom $\mathrm{PRM}_{C,\Lambda}$ expresses the fact that if each circuit in C computes an injective map (this is formulated as $\wedge_{i=1}^{t}(C^{(i)}(\overline{W_1}) \equiv C^{(i)}(\overline{W_2}) \to \overline{W_1} \equiv \overline{W_2})$), and if the images of the maps computed by each pair of circuits in C are disjoint (this is formulated as $\wedge_{1\le i<j\le t} C^{(i)}(\overline{W_1}) \not\equiv C^{(j)}(\overline{W_2})$), then we can assume that the assignments to the original variables X are taken from the image of C (this is formulated as $\vee_{i=1}^{t} C^{(i)}(\overline{W_1}) \equiv X$). The fact that the image of C is of size at least $2^n - \Lambda$ is expressed (due to Equation (7.1)) by the number of input bits (that is, $n-r$) of each circuit in C and the number of circuits in C (that is, t). Also note that the promise axiom is of polynomial-size as long as the circuits in C are (since $1/\varepsilon$ is a constant).

The following claim shows that the promise axioms are sound with respect to the promise Λ, in the sense that they do not discard too many truth assignments:

Claim: The promise axiom $\mathrm{PRM}_{C,\Lambda}$ discards at most Λ truth assignments. That is, there are at most Λ distinct assignments $\overline{a}$ to the X variables such that $\mathrm{PRM}_{C,\Lambda} \models X \not\equiv \overline{a}$.

Proof of claim: Assume that some Boolean map computed by some circuit in C is not injective. Then any assignment to the X variables has an extension ρ (to the extension variables in the promise axiom) that falsifies the premise of the main implication in $\mathrm{PRM}_{C,\Lambda}$ and thus ρ satisfies $\mathrm{PRM}_{C,\Lambda}$. Therefore no assignments to X are discarded.

Similarly, assume that the images of some pair of maps computed by two circuits in C are not disjoint. Then, again, any assignment to the X variables has an extension that satisfies $\mathrm{PRM}_{C,\Lambda}$, and so no assignments to X are discarded.

Assume that all the Boolean maps computed by circuits in C *are* injective and have pairwise disjoint images. Then every assignment satisfies the premise of the main implication in the promise axiom $\mathrm{PRM}_{C,\Lambda}$. Therefore, it suffices to show that the consequence of the main implication of the axiom (that is, $\vee_{i=1}^{t} C^{(i)}(\overline{W_1}) \equiv X$) discards at most Λ assignments to the X variables. By definition (of the encoding of the circuits) for all assignments $\overline{a}$ to the X variables that are in $\mathsf{Im}(C)$ there is an extension of $\overline{a}$ that satisfies $\vee_{i=1}^{t} C^{(i)}(\overline{W_1}) \equiv X$. Now, all the circuits $C^{(i)}$ compute injective maps with pairwise disjoint images, and thus by Equation (7.1) there are at least $2^n - \Lambda$ distinct elements (that is, assignments) in $\mathsf{Im}(C)$. Hence, at least $2^n - \Lambda$ assignments to the X variables are not discarded. $\blacksquare_{\text{Claim}}$

7.2.1.2 Smaller Promise

We now formulate promise axioms for promises smaller than $\varepsilon \cdot 2^n$. Specifically, we shall work with a promise of $\Lambda = 2^{\delta n}$ for a constant $0 < \delta < 1$ (we fix this Λ throughout this subsection). For such a promise, the promise axiom is similar to Definition 7.2.3, except that the number of input bits of each circuit in C needs to be modified accordingly. (We shall use the same terminology as that used above for the Big Promise.)

Definition 7.2.4 (Promise Axiom for $\Lambda = 2^{\delta n}$) *Let the promise be $\Lambda = 2^{\delta n}$, for a constant $1 < \delta < 1$, and let $t = \lceil(1-\delta)n\rceil$. Let C be a sequence of Boolean circuits $C := (C^{(1)}, \dots, C^{(t)})$. Assume that for each $1 \le i \le t$ the circuit $C^{(i)}$ has $n - i$ input bits and n output bits. Let $\overline{W_1}, \dots, \overline{W_t}$ and $\overline{W'_1}, \dots, \overline{W'_t}$ be $2t$ disjoint sets of extension variables*[1]*, where for all $1 \le i \le t$, W_i, W'_i consist of $n - i$ variables each. The promise axiom $\mathrm{PRM}_{C,\Lambda}$ is the CNF encoding (via the encoding defined in Section 7.5.1) of the following Boolean formula:*

$$\left(\bigwedge_{i=1}^{t} \left(C^{(i)}(\overline{W_i}) \equiv C^{(i)}(\overline{W'_i}) \to \overline{W_i} \equiv \overline{W'_i} \right) \wedge \bigwedge_{1 \le i < j \le t} C^{(i)}(\overline{W_i}) \not\equiv C^{(j)}(\overline{W_j}) \right) \longrightarrow \bigvee_{i=1}^{t} C^{(i)}(\overline{W_i}) \equiv X.$$

Note that the promise axiom is of polynomial size as long as the circuits in C are (since $t \le n$).

Also note that the proof of Claim 7.2.1.1 did not use the parameters r and t (which determine the number of input bits in the circuits in C and the number of circuits in C, respectively) but only the size $|\mathsf{Im}(C)|$. Thus, the same claim holds also for the promise axiom in Definition 7.2.4, which means that this promise axiom discards at most $2^n - |\mathsf{Im}(C)|$ truth assignments, for some sequence of circuits in C that compute injective maps with pairwise disjoint images. Therefore, we need to verify that $|\mathsf{Im}(C)| \ge 2^n - \Lambda$, for all C that consists of circuits computing injective maps with pairwise disjoint images.

Notice that for all $1 \le i \le t$ the circuit $C^{(i)}$ computes a Boolean map, denoted f_i, such that $f_i : \{0,1\}^{n-i} \to \{0,1\}^n$. Assume that all the f_i's are injective and that the images of each pair of functions f_i, f_j, for $1 \le i \ne j \le t$, are disjoint. Then, we have:

$$\begin{aligned} |\mathsf{Im}(C)| &= \left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^t} \right) \cdot 2^n = \left(1 - \frac{1}{2^t} \right) \cdot 2^n \\ &= 2^n - 2^{n - \lceil (1-\delta)n \rceil} \ge 2^n - 2^{\delta n} = 2^n - \Lambda \end{aligned}$$

Also note that $|\mathsf{Im}(C)| \le 2^n - 2^{\delta n - 1}$ and so if the circuit in C are injective with pairwise disjoint images then $\mathrm{PRM}_{C,\Lambda}$ discards *at least* $2^{\delta n}/2$ truth assignments.

[1]We have not been very economical in adding extension variables here; but this is not essential.

7.2.2 Promise Resolution

Definition 7.2.5 (Promise resolution) *Let Λ be the promise (see Definition 7.2.1) and let K be a CNF in the X variables. A* promise resolution (under the promise Λ) proof of the clause D from a CNF formula K *is a sequence of clauses $D_1, D_2, \ldots, D_\ell$ such that:*

(1) Each clause D_j is either a clause of K or a clause of a promise axiom $\mathrm{PRM}_{C,\Lambda}$ *(where* $\mathrm{PRM}_{C,\Lambda}$ *is either a big or a smaller promise axiom as defined in Definitions 7.2.3 and 7.2.4 and C is an arbitrary sequence of circuits with the prescribed input and output number of bits), or a resolvent of two previous clauses in the sequence, or a consequence of the weakening rule applied to one of the previous clauses in the sequence;*

(2) The sequence contains (the clauses of) at most one promise axiom;

(3) The last clause $D_\ell = D$.

The size, width *and* refutations *of promise resolution is defined the same as in resolution.*

Note that promise resolution is a Cook-Reckhow proof system: it is possible to efficiently verify whether a given CNF is an instance of the promise axiom, and hence to verify whether a sequence of clauses constitute a legitimate promise refutation. This can be done by "decoding" the CNF that encodes the promise axiom $\mathrm{PRM}_{C,\Lambda}$ and then checking that each circuit in C has the right number of input and output bits (we discuss this issue in some more detail in the appendix Section 7.5.1).

Proposition 7.2.6 *Let Λ be the promise (where Λ is either $\varepsilon \cdot 2^n$ or $2^{\delta n}$, for $0 < \varepsilon, \delta < 1$). Then promise resolution under the promise Λ is a sound and complete proof system for the set of CNF formulas under the promise Λ (see Definition 7.2.1). In other words, every unsatisfiable CNF has a promise resolution refutation and every CNF that has more than Λ satisfying assignments does not have promise resolution refutations.*

Proof. Completeness stems from completeness of resolution. Soundness under the promise Λ stems from Claim 7.2.1.1 (which, by the notes after Definition 7.2.4, holds for both the big and the smaller promise axioms). □

7.2.3 Discussion

Let K be an unsatisfiable CNF formula in n variables, $\mathrm{PRM}_{C,\Lambda}$ a promise axiom (where the circuits in C all compute injective and pairwise disjoint Boolean maps) and let $S := \mathsf{Im}(C) \subseteq \{0,1\}^n$ (such that $|S| \geq 2^n - \Lambda$). Then, one can think of a promise resolution refutation of K using the axiom $\mathrm{PRM}_{C,\Lambda}$ as containing two separate parts:

(i) a resolution 'refutation' of K where the space of truth assignments is restricted to S;

(ii) a resolution proof that $|S| \geq 2^n - \Lambda$.

Note that if we want to consider promise resolution as having only part (i), then we can modify (actually, strengthen) the promise axiom into $\vee_{i=1}^{t} C^{(i)}(\overline{W}) \equiv X$. However, this choice means losing the soundness of the proof system under the promise (that is, the soundness with respect to CNF formulas under a promise as defined in Definition 7.2.1), since we do not have any guarantee that the circuit C discards at most Λ assignments (and so CNF formulas with more than Λ satisfying assignments might have refutations in such a system).

It is possible to use any number of axioms of the form $C^{(i)}(\overline{W}) \equiv X$, as long as resolution can prove both the injectivity of each of the maps computed by the circuits $C^{(i)}$ introduced and the pairwise disjointness of these maps (as formulated by a propositional formula similar to the formulation in the promise axioms), and provided that the circuits $C^{(i)}$ have number of input bits that induce the right size of domains (that is, that the total size of their domains is at least $2^n - \Lambda$).

It is also possible to modify the promise axioms to suit any chosen size of promise Λ (possibly, only an approximation of Λ). This can be achieved by choosing a sequence of circuits with the appropriate size of domain (explicitly expressed by the number of input bits in each circuit in the sequence, and the total number of circuits).

Some comments about the formulation of the promise axioms are in order.

Comment 1: Note that we could not use only a single circuit C in the promise axioms (in contrast to a sequence of circuits), because that way we would not have the possibility of controlling the size of the domain of C and efficiently verifying that this size is the correct one inside resolution. To see this, note that if the number of input variables to C is n (the number of original variables) and the map computed by C is (provably) injective then C does not discard any assignment. If, on the other hand, the number of input variables to C is less than n, then C discards at least half the truth assignments, which might be too many.

Comment 2: Also note that in order to discard assignments we cannot use a seemingly more natural axiom of the form $C(\overline{W}) \not\equiv X$ for some circuit C (with domain of size Λ). The reason is that this would not discard assignments in the image of C: it is not necessarily true that $C(\overline{W}) \not\equiv X \models X \not\equiv \overline{b}$ for all $\overline{b} \in \{0,1\}^n$ such that $b \in \mathsf{Im}(C)$ (notice that even for such a $\overline{b}$ there might be some assignment $\overline{a}$ for which $C(\overline{a}) \not\equiv \overline{b}$).

On the other hand, Jan Krajíček notes (Krajíček (2007)) that it is possible to discard assignments by an axiom of the form $C(\overline{W}) \not\equiv X$, where C is a fixed circuit with domain of size at most Λ (that is, it has $k < n$ number of input bits, where $2^k \leq \Lambda$), and where the rule of using this axiom is that we can introduce any instance of $C(\overline{W}) \not\equiv X$ where *all the variables in* $\overline{W}$ *are substituted by constants* $0, 1$ *and variables from* X. This choice of axioms simplifies somewhat the actual formulation of the promise axioms, as it does not require that C computes an injective Boolean map. However, a possible drawback of such a formulation is the following: It is possible that for certain circuits (of the appropriate number of input and output bits) we shall need to use exponentially many such axiom instances to discard *all* (or most of) the assignments pertaining to the image of the circuits. In contrast to this, our formulation of the promise axioms above enables a single instance of a promise axiom using *any* circuit (more correctly, a sequence of circuits of the appropriate number of input and output bits) to discard *all* the assignments outside the image of the circuit.

7.3 Big Promise: Upper Bound

In this section, we show that under the promise $\Lambda = \varepsilon \cdot 2^n$, for any constant $0 < \varepsilon < 1$, resolution can efficiently certify the unsatisfiability of all unsatisfiable 3CNF formulas. The proof method resembles the algorithm presented by Trevisan in Trevisan (2004). For a constant k, this algorithm receives a kCNF formula K and deterministically approximates the fraction of satisfying assignments of K within an additive error of ε. The running time of the algorithm is linear in the size of K and polynomial in $1/\varepsilon$.

The idea behind the refutations in this section is based on the following observation: given a 3CNF formula K and a constant c, either there are $3(c-1)$ variables that hit[2] all the clauses in K or there are at least c clauses in K over $3c$ *distinct* variables denoted by K' (that is, each variable in K' appears only once). In the first case, we can consider all the possible truth assignments to the $3c$ variables inside resolution: if K is unsatisfiable then any such truth assignment yields an unsatisfiable 2CNF formula, which can be efficiently refuted in resolution (cf. Cook (1971)). In the second case, we can make use of a promise axiom to efficiently refute K' (this set of clauses has less then Λ satisfying assignments, for sufficiently large c). Specifically, in the second case, we construct a sequence of small circuits C for which any satisfying assignment for K' is *provably in resolution* (with polynomial-size proofs) outside the image of C.

The following is the main result of this section:

Theorem 7.3.1 *Let $0 < \varepsilon < 1$ be a constant and let $\Lambda = \varepsilon \cdot 2^n$ be the given promise. Then* every *unsatisfiable 3CNF with n variables has a polynomial-size (in n) resolution refutation under the promise Λ.*

This theorem is a consequence of the three lemmas that follow.

Lemma 7.3.2 *Let K be a 3CNF formula. For every integer c one of the following holds: (i) there is a set of at most $3(c-1)$ variables that hit all the clauses in K; or (ii) there is a sub-collection of clauses from K, denoted K', with at least c clauses and where each variable appears only once in K'.*

[2]A set of variables S that "hit all the clauses in a CNF formula K" is a set of variables for which every clause in K contains some variable from S.

Proof. Assume that $c > 2$ (otherwise the lemma is trivial). Suppose that there is no set of at most $3(c-1)$ variables that hit all the clauses in K and let D_1 be some clause in K. Then, there ought to be a clause D_2 from K that contains 3 variables that are not already in D_1 (or otherwise, the 3 (distinct) variables in D_1 hit all the clauses in K, which contradicts the assumption). In a similar manner we can continue to add new clauses from K until we reach a set of c clauses $D_1, D_2, \ldots, D_c$, where no variable appears more than once in this set of clauses. □

If case (i) of the prior lemma holds, then the following lemma suffices to efficiently refute the 3CNF:

Lemma 7.3.3 *Let c be constant and K be an unsatisfiable 3CNF formula in the X variables (where $n = |X|$). Assume that there is a set $S \subseteq X$ of at most $3(c-1)$ variables that hit all the clauses in K. Then there is a polynomial-size (in n) resolution refutation of K.*

Proof sketch: We simply run through all truth assignments to the variables in S (since $|S| \leq 3(c-1)$, there are only constant number of such truth assignments). Under each truth assignment to the S variables, K becomes an unsatisfiable 2CNF. It is known that any unsatisfiable 2CNF has a polynomial-size resolution refutation (cf. Cook (1971)). Thus, we can refute K with a polynomial-size resolution refutation. ■

If case (ii) in Lemma 7.3.2 holds, then it suffices to show that resolution under a big promise can efficiently refute any 3CNF formula T with a constant number of clauses (for a sufficiently large constant), where *each variable in T occurs only once* (such a T is of course satisfiable, but it has less than an ε fraction of satisfying assignments for a sufficiently large number of clauses). This is established in the following lemma.

Lemma 7.3.4 *Fix the constant $c = 3\lceil \log_{7/8}(\varepsilon/2) \rceil$. Let $\Lambda = \varepsilon \cdot 2^n$, where $0 < \varepsilon < 1$ is a constant and n is sufficiently large. Assume that T is a 3CNF with $c/3$ clauses (and c variables) over the X variables, where each variable in T occurs only once inside T. Then there is a polynomial-size resolution refutation of T under the promise Λ.*

Proof. The proof consists of constructing a sequence of polynomial-size circuits C (where the parameters of the circuits in C are taken from Definition 7.2.3; that is, $r = \lceil \log(1/\varepsilon) \rceil$ and $t = 2^r - 1$), such that: (i) resolution can efficiently prove the injectivity and the pairwise disjointness of the images of the circuits in C; and (ii) there is a polynomial-size refutation of T and $\mathrm{PRM}_{\Lambda,C}$. In other words, there is a polynomial-size derivation of the empty clause from the clauses of both T and $\mathrm{PRM}_{\Lambda,C}$.

Without loss of generality we assume that the variables in T are $x_1, \ldots, x_c$. The sequence C consists of the circuits $C^{(1)}, \ldots, C^{(t)}$, where each circuit $C^{(i)}$ has $n-r$ input bits and n output bits. Denote the Boolean circuit that computes the jth output bit of $C^{(i)}$ by $C_j^{(i)}$ and let the input variables of all the circuits in C be $\overline{W} := \{w_1, \ldots, w_{n-r}\}$. As shown in equation (7.1), since the circuits in C are intended to compute injective and pairwise image-disjoint maps, the image of C would be of size $2^n - 2^{n-r}$. We now define the map that each circuit in C computes.

First, we determine the Boolean functions computed by the output bits in positions $c+1, \dots, n$ in all the circuits in C. For all $1 \le i \le t$ and all $c+1 \le j \le n$ let $C_j^{(i)}(\overline{W})$ compute the $(j-r)$th input variable w_{j-r}.

Second, we need to determine the rest of the output bits for all the circuits in C, that is, we need to determine the Boolean functions computed by $C_j^{(i)}$, for all $1 \le i \le t$ and all $1 \le j \le c$. Our intention is that for all $1 \le i \le t$, the (single output) circuits $C_1^{(i)}, \dots, C_c^{(i)}$ should compute (when combined together) a Boolean map, denoted by f_i, from $c-r$ input bits $\overline{W}_0 := \{w_1, \dots, w_{c-r}\}$, to c output bits. The jth output bit of f_i (which is computed by $C_j^{(i)}$) is denoted by $f_{i,j}$, for $1 \le j \le c$. In other words, $f_i(\overline{W}_0) = f_{i,1}(\overline{W}_0) \circ \cdots \circ f_{i,c}(\overline{W}_0)$, where $\circ$ denotes concatenation of bits (we shall describe the functions f_i below). Summing it up for now, we have the following:

$$\begin{array}{l} C_1^{(1)}(\overline{W}_0) = f_{1,1}(\overline{W}_0), \ \dots, \ C_c^{(1)}(\overline{W}_0) = f_{1,c}(\overline{W}_0), \\ \qquad C_{c+1}^{(1)}(w_{c-r+1}) = w_{c-r+1}, \ \dots, \ C_n^{(1)}(w_{n-r}) = w_{n-r} \\ \qquad \vdots \qquad\qquad\qquad\qquad \vdots \\ C_1^{(t)}(\overline{W}_0) = f_{t,1}(\overline{W}_0), \ \dots, \ C_c^{(t)}(\overline{W}_0) = f_{t,c}(\overline{W}_0), \\ \qquad C_{c+1}^{(t)}(w_{c-r+1}) = w_{c-r+1}, \ \dots, \ C_n^{(t)}(w_{n-r}) = w_{n-r}, \end{array} \tag{7.2}$$

where $C_j^{(i)}(w_k) = w_k$ denotes the fact that $C_j^{(i)}$ outputs the (input) variable w_k (in which case we assume that the circuit $C_j^{(i)}$ consists of only a single gate: the variable w_k); and where $C_j^{(i)}(\overline{W}_0) = f_{i,j}(\overline{W}_0)$ denotes the fact that $C_j^{(i)}$ computes the function $f_{i,j}$ in the $c-r$ input variables $\overline{W}_0$.

We now describe the requirements of the functions $f_{i,j}$. Specifically, let $B \subseteq \{0,1\}^c$ be the set of all *falsifying* assignments[3] to T and denote by $\mathsf{Im}(f_i)$ the image of f_i, for all $1 \le i \le t$. We need the f_i's functions to map every input (over $c-r$ input bits) to a truth assignment (over the c variable $x_1, \dots, x_c$) that *falsifies* T (that is, a truth assignment from B). We also need the f_i's to be injective and have pairwise disjoint images by the requirements of the promise axiom (Definition 7.2.3). So that, overall, the Boolean maps computed by the circuits in C would *discard all the truth assignments that satisfy* T. To prove the existence of such f_i's we need the following proposition:

Proposition 7.3.5 *There exists a collection of Boolean functions f_i, where $1 \le i \le t$, for which the following three properties hold.*

1. $\cup_{i=1}^{t} \mathsf{Im}(f_i) \subseteq B$;

2. *All the f_i's are injective and have pairwise disjoint images;*

[3]Note that the assignments here are actually *partial* truth assignments with respect to X, that is, they give truth values only to the variables $x_1, \dots, x_c$ (these are all the variables in T).

3. *All the f_i's depend on a constant number of input variables: $w_1, \ldots, w_{c-r}$ (and hence, all the $f_{i,j}$'s depend only on these variables).*

Proof. Each f_i should depend on $c-r$ variables and should be injective, and further, each pair of f_i's should have disjoint images; thus we have:

$$\left|\bigcup_{i=1}^{t} \mathsf{Im}(f_i)\right| = t \cdot 2^{c-r} = (2^r - 1) \cdot 2^{c-r} = 2^c \cdot (1 - 2^{-r})\,. \tag{7.3}$$

Hence, to prove the existence of the collection of f_i's with the required three properties it suffices to show that $|\cup_{i=1}^{t} \mathsf{Im}(f_i)| \le |B|$, and by (7.3) it suffices to show:

$$2^c \cdot (1 - 2^{-r}) \le |B|\,. \tag{7.4}$$

Observe that the fraction of distinct assignments that satisfy T is equal to the probability (over all truth assignments to T) that a uniformly chosen random truth assignment satisfies all the $c/3$ clauses in T, which is equal to

$$\left(\frac{7}{8}\right)^{c/3} = \left(\frac{7}{8}\right)^{\lceil \log_{7/8}(\varepsilon/2) \rceil}, \tag{7.5}$$

and so

$$|B| = 2^c \cdot \left(1 - \left(\frac{7}{8}\right)^{\left\lceil \log_{7/8}(\varepsilon/2) \right\rceil}\right).$$

Therefore, for (7.4) to hold it remains to show

$$2^{-r} \ge \left(\frac{7}{8}\right)^{\left\lceil \log_{7/8}(\varepsilon/2) \right\rceil},$$

which holds because

$$\begin{aligned} 2^{-r} &= 2^{-\lceil \log(1/\varepsilon) \rceil} \ge 2^{-\log(1/\varepsilon) - 1} = 2^{-\log(1/\varepsilon)}/2 \\ &= \varepsilon/2 = \left(\tfrac{7}{8}\right)^{\log_{7/8}(\varepsilon/2)} \ge \left(\tfrac{7}{8}\right)^{\left\lceil \log_{7/8}(\varepsilon/2) \right\rceil}. \end{aligned}$$

□

Having established the existence of functions f_i for which the three conditions in Proposition 7.3.5 hold, we define each circuit $C_j^{(i)}$, for $1 \le i \le t$ and $1 \le j \le c$, to compute the function $f_{i,j}$. Since the domain of each $f_{i,j}$ is constant (that is, 2^{c-r}) *each $C_j^{(i)}$ can be of constant size.*

Note that the circuits in $C = (C^{(1)}, \ldots, C^{(t)})$ indeed compute t injective Boolean maps that have pairwise disjoint images. Disjointness of images stems from the fact that the f_i's functions all have disjoint images, and injectivity stems from the fact that the f_i's are all injective, and that for each $1 \le i \le t$, the Boolean map computed by $C_{c+1}^{(i)} \circ \ldots \circ C_n^{(i)}$ (again, $\circ$ denotes concatenation of bits) is exactly the identity map $\mathrm{id}\colon \{0,1\}^{n-c} \to \{0,1\}^{n-c}$.

To complete the proof of Lemma 7.3.4, we need to show that *resolution can efficiently prove* that indeed the circuits in C all compute injective Boolean maps and have pairwise disjoint images (as well as to efficiently refute T when assuming that X can take only assignments from the image of C). This is done in the following claim:

Claim: Let C be the sequence of circuits as devised above, and let $\mathrm{PRM}_{C,\Lambda}$ be the corresponding (big) promise axiom. Then there is a polynomial-size resolution refutation of T and $\mathrm{PRM}_{C,\Lambda}$.

Proof of claim: The proof follows by considering the encoding of the (big) promise axiom $\mathrm{PRM}_{C,\Lambda}$ via the encoding scheme in the appendix (Section 7.5.1) and showing how resolution can prove the empty clause from T and this encoding. Here we shall use a less formal description; more details can be found in the appendix.

First we need resolution to prove the premise of the main implication in $\mathrm{PRM}_{C,\Lambda}$. This breaks into two parts corresponding to $\wedge_{i=1}^{t}\left(C^{(i)}(\overline{W_1}) \equiv C^{(i)}(\overline{W_2}) \rightarrow \overline{W_1} \equiv \overline{W_2}\right)$ and $\wedge_{1\le i<j\le t} C^{(i)}(\overline{W_1}) \not\equiv C^{(j)}(\overline{W_2})$.

For the first part, we need to refute the statement expressing that C contains some circuit $C^{(i)}$ that computes a non-injective Boolean map. This can be efficiently refuted in resolution: Assume (inside resolution) that $C^{(i)}(\overline{W_1}) \equiv C^{(i)}(\overline{W_2})$, for some $1 \le i \le t$, then by (7.2) we can efficiently prove (inside resolution) that for all $c-r+1 \le j \le n-r$ it happens that $w_j^{(1)} \equiv w_j^{(2)}$ (where $w_j^{(1)}$ is the jth variable in $\overline{W}_1$, and $w_j^{(2)}$ is the jth variable in $\overline{W}_2$) (see details in the appendix, and in particular Section 7.5.1.3). Thus, it remains to refute the statement that for some $1 \le j \le c-r$ it happens that $w_j^{(1)} \not\equiv w_j^{(2)}$. This is indeed a contradiction by definition of the circuits in C (as they compute injective maps). Since all the output bits $w_j^{(1)}, w_j^{(2)}$ for $1 \le j \le c-r$, are computed by constant size circuits $C_j^{(i)}$ for $1 \le j \le c-r$ and $1 \le i \le t$ (with constant number of input variables), such a contradiction can be refuted in constant size resolution refutation (again, see more details in the appendix).

The disjointness of the images of the (maps computed by the) circuits in C is also efficiently provable inside resolution in a similar manner, and we shall not describe it here.

Therefore, we arrive (inside resolution) at the consequence of the main implication in the promise axiom: $\vee_{i=1}^{t} C^{(i)}(\overline{W}_1) \equiv X$. It remains to refute T and $\vee_{i=1}^{t} C^{(i)}(\overline{W}_1) \equiv X$.

Again, T has a constant number of variables (that is, c). Consider only the circuits that output to the variables $x_1, \ldots, x_c$ in $\vee_{i=1}^{t} C^{(i)}(\overline{W}_1) \equiv X$: these are the circuits $C_1^{(i)}, \ldots, C_c^{(i)}$ for all $1 \le i \le t$. We shall denote the set of these circuits by C'. The (functions computed by) the circuits in C' depend on a constant number of variable $\overline{W}_0$ and they have constant size. Denote by Z the subformula of $\mathrm{PRM}_{C,\Lambda}$ that contains the (encoding of) the circuits in C' including the encoding of the statement that for some $1 \le i \le t$ the variables $x_1, \ldots, x_c$ are equal to the output of the circuits $C_1^{(i)}, \ldots, C_c^{(i)}$. By the definition of the circuits in C (see condition (1) in Proposition 7.3.5) Z discards all the satisfying assignments of T (over the c variables in T)[4]. Thus, T and Z constitute

[4]We have abused notation here, as we defined *assignment discarding* of only *complete assignments to X*

together a contradiction of constant size (as there are no satisfying assignments for both T and Z). Therefore, there is a constant size resolution refutation of T and Z. $\blacksquare_{\text{Claim}}$

This concludes the proof of Lemma 7.3.4. $\square$

7.4 Smaller Promise: Lower Bound

In this section, we prove an exponential lower bound on the size of resolution refutations under the promise $2^{\delta n}$, for any constant $0 \leq \delta \leq 1$. The lower bound apply to random 3CNF formulas with $O(n^{3/2-\epsilon})$ number of clauses, for $0 < \epsilon < \frac{1}{2}$ (where n is the number of variables in the 3CNF). This lower bound matches the known lower bound on resolution refutation-size for random 3CNF formula (without any promise). Basically, the proof strategy of our lower bound is similar to that of Ben-Sasson and Wigderson (2001), except that we need to take care that every step in the proof works with the augmented (smaller) promise axiom.

The lower bound is somewhat stronger than described above in two respects. First, we show that restricting the set of all truth assignments 2^n to *any* smaller set (that is, not just those sets defined by small circuits) that consists of $2^n - 2^{\delta n}$ assignments (for any constant $0 \leq \delta \leq 1$), does not give resolution any advantage in the average-case. One can think of such a restriction as modifying the semantic implication relation $\models$ to take into account only assignments from some prescribed set of assignments S, such that $|S| = 2^n - 2^{\delta n}$ (in other words, for two formulas A, B, we have that $A \models B$ under the restriction to S iff any truth assignment *from* S that satisfies A also satisfies B). Formally, this means that the lower bound does not use the fact that the restricted domain of size $2^n - 2^{\delta n}$ is defined by a sequence C of polynomial-size circuits (nor the fact that the circuits in C ought to have polynomial-size resolution proofs of their injectivity and pairwise disjointness).

Second, we could allow for a promise that is bigger than $2^{\delta n}$, and in particular for a promise of $2^{n(1-1/n^{1-\xi})} = 2^n/2^{n^\xi}$, for some constant $0 < \xi < 1$ (see remark after Theorem 7.4.2 below). The actual proof of the lower bound uses the smaller promise of $2^{\delta n}$, but the proof for a $2^n/2^{n^\xi}$ promise is the same. (Although we have not defined precisely how the promise axioms are formulated in the case of a promise equal to $2^n/2^{n^\xi}$, it is possible to formulate such promise axioms along the same lines described in Definition 7.2.4.)

The following defines the usual *average-case* setting of 3CNF formulas (there are other definitions that are essentially similar):

Definition 7.4.1 (Random 3CNF formulas) *For a* 3CNF *formula K with n variables X and $\beta \cdot n$ clauses, we say that β is the* density *of K. A* random 3CNF formula *on n variables and density β is defined by picking $\beta \cdot n$ clauses from the set of all $2^3 \cdot \binom{n}{3}$ clauses, independently and indistinguishably distributed, with repetitions.*

while here we say that Z discards a partial assignment to the variables $x_1, \ldots, x_c$ only. But the definition of such partial assignment discarding is similar (consider the variables $x_1, \ldots, x_c$ to be the only *original variables*).

We say that an event (usually a property of a 3CNF in n variables and density β) happens *with high probability* if it happens with $1 - o(1)$ probability in the specified probability space (usually random 3CNF formulas as defined in Definition 7.4.1).

Our goal is to prove a lower bound on the average-case refutation-size of 3CNF formulas taken from the *set of 3CNF formulas under a promise* as defined in Definition 7.2.1 (note that the probability space defined in Definition 7.4.1 is defined over a different set of 3CNF, that is, the set of *all* 3CNF formulas). For this purpose, we define a probability space over the set of 3CNF formulas under a promise: The distribution of *random* 3CNF *formulas under a promise* Λ on n variables and density β is the distribution of random 3CNF formulas in Definition 7.4.1 *conditioned* on the event that the 3CNF is either unsatisfiable or has more than $\Lambda(n)$ satisfying assignments.

We now argue that to satisfy our goal to prove a lower bound on the average-case proof complexity of 3CNF formulas under a promise, it is sufficient to prove the lower bound result considering the distribution of random 3CNF formulas as defined in Definition 7.4.1.

It is well known that almost all 3CNF formulas with a density β above a certain constant threshold (say, 5) are unsatisfiable. This means that any property of a 3CNF (with density above the threshold) that happens with high probability in the distribution in Definition 7.4.1 also happens with high probability in the distribution of random 3CNF formulas under a promise $\Lambda(n)$ (as defined above); this is because there are only a fraction $o(1)$ of 3CNF formulas (with a given fixed number of variables n and a given fixed density β above the threshold) that are satisfiable (and moreover have at least one satisfying assignment but less than $\Lambda(n)$ satisfying assignments). Thus, if we prove that with high probability a random 3CNF formula has no small promise resolution refutation then it implies also that with high probability *a random 3CNF formula under a promise* has no small promise resolution refutation. *Therefore, we shall consider from now on only the distribution of 3CNF formulas as defined in Definition 7.4.1, and forget about the other distribution.*

7.4.1 The Lower Bound

Throughout this section we fix $0 < \delta < 1$ and $\Lambda = 2^{\delta n}$. We also fix an arbitrary instance of a promise axiom $\mathrm{PRM}_{C,\Lambda}$ (from Definition 7.2.4; where C is a sequence of the appropriate number of circuits, and each circuit in C have the appropriate number of input and output bits). For K a CNF formula, we denote by $\mathrm{Vars}(K)$ the set of variables that occur in K.

The following is the main theorem of this section. The lower bound rate matches that appearing in Ben-Sasson and Wigderson (2001) for resolution.

Theorem 7.4.2 *Let $0 < \delta < 1$ and $0 < \epsilon < 1/2$. With high probability a random 3CNF formula with $\beta = n^{1/2-\epsilon}$ requires a size $\exp(\Omega(\beta^{-4/(1-\epsilon)} \cdot n))$ resolution refutation under the promise $\Lambda = 2^{\delta n}$.*

Remark: As mentioned above, we could allow in Theorem 7.4.2 for a promise that is bigger than $2^{\delta n}$, and precisely for a promise of $2^{n(1-\frac{1}{n^{1-\xi}})} = 2^n/2^{n^\xi}$, for any constant ξ

such that $\frac{\epsilon}{(1-\epsilon)} < \xi < 1$ (for instance, this allows for a promise of $2^n/2^{n^{1/3}}$).

The proof strategy of Theorem 7.4.2 is to show that with high probability for a random 3CNF formula K with density $\beta = n^{1/2-\epsilon}$, a resolution refutation under the promise $2^{\delta n}$ of K must contain some clause D of large width. Then we can apply the size-width tradeoff from Theorem 7.1.1 to reach the appropriate size lower bound.

In order to illustrate an exponential lower bound via the size-width tradeoff of Theorem 7.1.1, we need to guarantee that all the initial clauses (that is, all the axiom clauses) are of *constant width*. The 3CNF formula K is certainly of constant width, but the clauses pertaining to the promise axiom $\text{PRM}_{C,\Lambda}$ might not be (see the appendix Section 7.5.1 for a detailed specification of these clauses). We solve this problem as follows. First, we add yet more extension variables to encode the clauses of the promise axiom with new constant width clauses. Second, we note that the original clauses of the promise axiom can be derived by a linear-size resolution proof from the new constant width version of the promise axiom (therefore, if there is a polynomial-size resolution refutation of K using the original promise axiom, then there is also a polynomial-size resolution refutation of K using the new constant width version of the promise axiom). Finally, we prove the exponential lower bound on resolution augmented with the constant width version of the promise axiom (instead of the original clauses pertaining to the promise axiom).

We now explain how to get the new constant-width promise axiom from the clauses pertaining to the original promise axiom from Definition 7.2.4 (as depicted in the appendix). Let $E = \ell_1 \vee \ldots \vee \ell_m$ be a clause in the promise axiom that has more than constant width (that is, ℓ_i's are literals and $m = \omega(1)$). Then, we replace the clause E with the following collection of clauses:

$$\ell_1 \vee e_1,\ \neg e_1 \vee \ell_2 \vee e_2,\ \neg e_2 \vee \ell_3 \vee e_3,\ \ldots\ , \neg e_{m-1} \vee \ell_m, \tag{7.6}$$

where the e_i's are new extension variables. By resolving on the e_i variables, one after the other, it is possible to derive with a linear-size resolution proof the original clause E from the clauses in (7.6) (consider the first two clauses (from left) in (7.6), and resolve over the variable e_1, then the resolvent of this step is resolved over the variable e_2 with the third clause in (7.6), and so forth). (Note that every truth assignment that satisfies (7.6) also satisfies E, and so any clause that is semantically implied by E (see the preliminaries, Section 7.1) is also semantically implied by (7.6). This means that the new constant width version of the promise axiom discards the same truth assignments to the variables X as the original version of the promise axiom.)

Thus, *from now on in this section we assume that the promise axiom consists of clauses of a constant width.*

The rest of this section is devoted to the proof of Theorem 7.4.2.

For a clause D define:

$$\eta(D) := \min\left\{|K'| \;\middle|\; K' \subseteq K \text{ and } (\text{PRM}_{C,\Lambda} \cup K') \models D\right\}.$$

Remark: We use the symbol η to distinguish it from a similar measure μ used in Ben-Sasson and Wigderson (2001): Here we require the minimal set of clauses from K that *combined with the axiom* $\mathrm{PRM}_{C,\Lambda}$ semantically imply D.

We show that with high probability for a random 3CNF formula with density $\beta = n^{1/2-\epsilon}$, for $0 < \epsilon < 1/2$, the following is true:

1. Let $k = 2n \cdot (80\beta)^{-2/(1-\epsilon)}$. Then $\eta(\Box) \geq k$.
2. Any refutation of K must contain a clause D such that $k/2 \leq \eta(D) \leq k$.
3. Any clause D from 2 must have large width, and specifically $|D| \geq \epsilon n(80\beta)^{-2/(1-\epsilon)}$.

Note that by Theorem 7.1.1, Item 3 above concludes the proof of Theorem 7.4.2.

The following two definitions are similar to those in Ben-Sasson and Wigderson (2001) (we refer directly to 3CNF formulas instead of 3-uniform hypergraphs):

Definition 7.4.3 (CNF Expansion) *For a 3CNF formula K with density $\beta = n^{1/2-\epsilon}$, for $0 < \epsilon < 1/2$, the* expansion *of K is*

$$e(K) := \min \left\{ 2|\mathrm{Vars}(K')| - 3|K'| \;\middle|\; \begin{array}{l} K' \subset K \text{ and} \\ n \cdot (80\beta)^{-2/(1-\epsilon)} \leq |K'| \\ \leq 2n \cdot (80\beta)^{-2/(1-\epsilon)} \end{array} \right\}.$$

Definition 7.4.4 (Partial matchability) *A 3CNF formula K with density $\beta = n^{1/2-\epsilon}$, for $0 < \epsilon < 1/2$, is called* partially matchable *if for all $K' \subset K$ such that $|K'| \leq 2n \cdot (80\beta)^{-2/(1-\epsilon)}$ we have $|\mathrm{Vars}(K')| \geq |K'|$.*

The next lemma gives two properties of random 3CNF formulas that occur with high probability (see the appendix of Ben-Sasson and Wigderson (2001) for a proof). We then use this lemma to show that with high probability 1,2,3 above hold.

Lemma 7.4.5 (Beame et al. (2002)) *Let $0 < \epsilon < 1/2$ and let K be a random 3CNF with n variables and density $\beta = n^{1/2-\epsilon}$, then with high probability:*

1. *$e(K) \geq \epsilon n(80\beta)^{-2/(1-\epsilon)}$; and*
2. *K is partially matchable.*

We now prove 1. In light of part (2) in Lemma 7.4.5, in order to prove that with high probability 1 holds it is sufficient to prove the following:

Lemma 7.4.6 *Let K be a 3CNF formula in the X variables with density $\beta = n^{1/2-\epsilon}$, for $0 < \epsilon < 1/2$. If K is partially matchable then $\eta(\Box) \geq 2n \cdot (80\beta)^{-2/(1-\epsilon)}$.*

Proof. By partial matchability of K, for all $K' \subset K$ such that $|K'| \leq 2n \cdot (80\beta)^{-2/(1-\epsilon)}$ it happens that $|\mathrm{Vars}(K')| \geq |K'|$. Thus, by Hall's Theorem we can choose a distinct variable (representative) from each clause in K' and set it to satisfy its clause. Clearly, $|\mathrm{Vars}(K')| \leq 3|K'| \leq 6n \cdot (80\beta)^{-2/(1-\epsilon)}$, and so there is a (partial) truth assignment ρ to at most $6n \cdot (80\beta)^{-2/(1-\epsilon)}$ variables in X that satisfies K'. Since $\beta = n^{1/2-\epsilon}$,

$$6n \cdot (80\beta)^{-2/(1-\epsilon)} = 6 \cdot 80^{-2/(1-\epsilon)} \cdot n^{\epsilon/(1-\epsilon)}, \tag{7.7}$$

which, by $0 < \epsilon < 1/2$, is equal to $O(n^{\lambda})$ for some $0 < \lambda < 1$. Thus for sufficiently large n there are more than δn variables from X not set by ρ, which means that there are more than $2^{\delta n}$ different ways to extend ρ into truth assignments (to all the variables in X) that satisfy K'.[5] Since the promise axiom $\mathrm{PRM}_{C,\Lambda}$ can discard up to $2^{\delta n}$ truth assignments to the X variables, we get that $\mathrm{PRM}_{C,\Lambda} \cup K'$ is *satisfiable* (any assignment to X that is not discarded by $\mathrm{PRM}_{C,\Lambda}$ can be extended to the extension variables in a way that satisfies $\mathrm{PRM}_{C,\Lambda}$).

We have thus showed that every collection K' containing at most $2n \cdot (80\beta)^{-2/(1-\epsilon)}$ clauses from K and augmented with the promise axiom $\mathrm{PRM}_{C,\Lambda}$ is satisfiable. This implies in particular that $\eta(\Box) \geq 2n \cdot (80\beta)^{-2/(1-\epsilon)}$. □

We now prove 2. Note that the resolution rule is *sub-additive* with respect to η in the sense that for all three clauses E, F, D such that D is a resolvent of E and F, it holds that

$$\eta(E) + \eta(F) \geq \eta(D).$$

We also clearly have that for every axiom clause E (either from K or from the promise axiom):

$$\eta(E) = 1.$$

Let

$$k = 2n \cdot (80\beta)^{-2/(1-\epsilon)}.$$

By Lemma 7.4.6, with high probability for a 3CNF formula K with density $\beta = n^{1/2-\epsilon}$ (for $0 < \epsilon < 1/2$) it happens that $\eta(\Box) \geq k$. By sub-additivity of the resolution rule with respect to η, in any resolution refutation of K under the promise Λ, there ought to be some clause D such that

$$k/2 \leq \eta(D) \leq k.$$

We now prove 3. Let D be a clause such that $k/2 \leq \eta(D) \leq k$ from 2 and let K' be the (minimal) set of clauses from K for which $\mathrm{PRM}_{C,\Lambda} \cup K' \models D$ and $k/2 \leq |K'| \leq k$. We shall prove that (with high probability for a random 3CNF) $|D| \geq \epsilon n (80\beta)^{-2/(1-\epsilon)}$. In light of Lemma 7.4.5 part (1), in order to prove this, it is sufficient to prove the next two lemmas.

[5]Actually, for sufficiently large n there are more than $\Omega(n - n^{\epsilon/(1-\epsilon)})$ such variables, from which we can assume the bigger promise $\Lambda = 2^n/2^{n^{\xi}}$, for any $\frac{\epsilon}{(1-\epsilon)} < \xi < 1$, as noted in the remark after Theorem 7.4.2.

Define $\partial K'$, called the *boundary of* K', to be the set of variables in K' that occur only *once* in K' (in other words, each variable in $\partial K'$ appears only in one clause in K').

Lemma 7.4.7 $|\partial K'| \geq e(K)$.

Proof. Every variable not in $\partial K'$ must be covered by at least two distinct clauses in K', and so $|\mathrm{Vars}(K')| \leq |\partial K'| + \frac{1}{2} \cdot (3|K'| - |\partial K'|)$. Thus, we have $|\partial K'| \geq 2|\mathrm{Vars}(K')| - 3|K'| \geq e(K)$ (where the last inequality is by Definition 7.4.3 and since $k/2 \leq |K'| \leq k$). $\square$

Lemma 7.4.8 $|D| \geq |\partial K'|$.

Proof. Let $x_i \in \partial K'$, for some $1 \leq i \leq n$, and denote by K_i the (unique) clause from K' that contains x_i. Assume by a way of contradiction that x_i does not occur in D.

By minimality of K' with respect to η and D there exists an assignment α (here we treat α as a *total* truth assignment, that is, a truth assignment to both the X variables and the extension variables in the promise axiom) such that

$$(K' \setminus K_i)(\alpha) = 1 \ \text{ and } \ D(\alpha) = 0 \tag{7.8}$$

(as otherwise $(K' \setminus K_i) \models D$ which clearly implies $\mathrm{PRM}_{C,\Lambda} \cup (K' \setminus K_i) \models D$, which then contradicts the minimality of K' with respect to η and D).

By assumption, x_i occurs neither in $K' \setminus K_i$ nor in D. Hence, we can flip the value of α on x_i so that $K_i(\alpha) = 1$ while still keeping (7.8) true. We thus have:

$$K'(\alpha) = 1 \ \text{ and } \ D(\alpha) = 0 \tag{7.9}$$

Since $|K'| \leq k$, we have that $|\mathrm{Vars}(K')| \leq 3k = 6n \cdot (80\beta)^{-2/(1-\epsilon)}$ (recall that $|K'|$ is the number of *clauses* in K'). If $|D| \geq |\partial K'|$ we are done. Otherwise,

$$|\mathrm{Vars}(K')| + |D| < |\mathrm{Vars}(K')| + |\partial K'| \leq 2|\mathrm{Vars}(K')| \leq 12n \cdot (80\beta)^{-2/(1-\epsilon)}.$$

Thus, similar to equation (7.7), for sufficiently large n, the total number of distinct variables in K' and D is at most $|\mathrm{Vars}(K')| + |D| = O(n^{\lambda})$, for some $0 < \lambda < 1$. This means that for sufficiently large n there are more than δn variables from X for which flipping the value of α on them still validates (7.9).[6] Hence, there are more than $2^{\delta n}$ distinct assignments to the X variables for which (7.9) holds.

The promise axiom $\mathrm{PRM}_{C,\Lambda}$ discards at most $2^{\delta n}$ assignments to the X variables. This means that there are at most $2^{\delta n}$ assignments ρ to the X variables that falsify $\mathrm{PRM}_{C,\Lambda}$ (that is, that every extension of ρ to all the extension variables falsifies $\mathrm{PRM}_{C,\Lambda}$), while *all* other assignments ρ to the X variables have an extension (to all the extension variables) that *satisfies* $\mathrm{PRM}_{C,\Lambda}$. Thus, by the previous paragraph there ought to be at least one

[6]Again, similar to what was noted in the proof of Lemma 7.4.6, for sufficiently large n there are actually more than $\Omega(n - n^{\epsilon/(1-\epsilon)})$ such variables.

assignment ρ to the X variables that has an extension ρ' to the extension variables, such that

$$\mathrm{PRM}_{C,\Lambda}(\rho') = 1\,, K'(\rho') = 1 \text{ and } D(\rho') = 0, \tag{7.10}$$

which contradicts the assumption that $\mathrm{PRM}_{C,\Lambda} \cup K' \models D$. □

7.5 Appendix: Encodings

7.5.1 Encoding of Boolean Circuits and Promise Axioms

In this section we describe in detail how the promise axiom (see Definition 7.2.3) is encoded as a CNF formula. As mentioned in Section 7.1.2, by Cook's Theorem there is always an efficient way to encode a Boolean circuit as a CNF formula using extension variables (that is, a CNF with size $O(s \cdot \log(s))$ can encode a circuit of size s). However, we shall need to be more specific regarding the actual way the encoding is done since we require that resolution should be able to efficiently prove some basic facts about the encoded circuits.

7.5.1.1 Boolean Circuit Encoding

The following definition is similar to the *circuit encoding* defined in Alekhnovich *et al.* in Alekhnovich et al. (2004) (note that it deals with a *single output bit* Boolean circuit):

Definition 7.5.1 (Encoding of Boolean circuits) *Let $C(\overline{W})$ be a Boolean circuit (with $\vee, \wedge$ as fan-in two gates and $\neg$ a fan-in one gate) and m input variables $\overline{W} := w_1, \ldots, w_m$ and a* single output bit. *For every gate v of the circuit C we introduce a special extension variable y_v. For input gates w_j ($1 \le j \le m$) we identify y_{w_j} with w_j. We denote by y^1 the literal y and by y^0 the literal $\neg y$. The CNF formula $\|C(\overline{W})\|$ consists of the following clauses:*

(i) $y_{v_1}^{\bar{\epsilon}_1} \vee y_{v_2}^{\bar{\epsilon}_2} \vee y_v^{\pi_\circ(\epsilon_1,\epsilon_2)}$, where v is a $\circ \in \{\vee, \wedge\}$ gate in C and v_1, v_2 are the two input gates of v in C and $\langle \epsilon_1, \epsilon_2 \rangle$ is any vector in $\{0,1\}^2$ and $\pi_\circ$ is the truth table function of $\circ$ (and $\bar{0} = 1,\ \bar{1} = 0$);

(ii) $y_{v_1}^{\bar{\epsilon}_1} \vee y_v^{\pi_\neg(\epsilon_1)}$, where v is a $\neg$ gate in C and v_1 is the single input gates of v in C, and $\epsilon_1 \in \{0,1\}$ and $\pi_\neg$ is the truth table function of $\neg$.

We write $\|C(\overline{W})\|(y)$ to indicate explicitly that the output gate v of C is encoded by the extension variable y.

7.5.1.2 Encoding of the Promise Axioms

We now give a rather detailed description of how the promise axioms are encoded as CNF formulas. We shall consider only the 'big' promise axiom (Definition 7.2.3), but the other variant (Definition 7.2.4) is similar. We encode the promise axioms in a bottom-up manner, encoding the sub-formulas separately, and then combining all of them together.

We assume that a Boolean circuit $C(\overline{W})$ with n output bits is encoded as n distinct circuits and we write $\|C(\overline{W})\|(\overline{Y})$ to indicate explicitly that the output gates $v_1, \ldots, v_n$ of C are encoded by the extension variables $y_1, \ldots, y_n$ (where $\overline{Y} := \{y_1, \ldots, y_n\}$). This means that $\|C(\overline{W})\|(\overline{Y})$ is the CNF formula $\wedge_{i=1}^{n}\|C_i(\overline{W})\|(y_i)$, where $C_i(\overline{W})$ is the circuit computing the ith output bit of $C(\overline{W})$ and y_i is the variable that encodes (see Definition 7.5.1) the (single) output bit of $C_i(\overline{W})$. We also require that if the (function computed by the) circuit $C(\overline{W})$ does not depend,[7] on some input bit w_i, then w_i does not occur in the encoding of $C(\overline{W})$.

Let $1 \leq k \leq t$ (where the parameter t is taken from Definition 7.2.3). We first encode as a CNF formula the negation of following sub-formula from the promise axiom:

$$C^{(k)}(\overline{W}_1) \equiv C^{(k)}(\overline{W}_2) \ \rightarrow \overline{W}_1 \equiv \overline{W}_2\,.$$

We denote this CNF encoding by $\neg\mathrm{INJ}_k$ (where INJ stands for *injective*).

Definition 7.5.2 ($\neg$INJ$_k$) *Let $1 \leq k \leq t$ and $m = n - r$ (all the parameters are taken from Definition 7.2.3). Let $\overline{W}_1 := \{w_1^{(1)}, \ldots, w_m^{(1)}\}$, $\overline{W}_2 := \{w_1^{(2)}, \ldots, w_m^{(2)}\}$, $\overline{Y}_k := \{y_1^{(k)}, \ldots, y_n^{(k)}\}$ and $Z_k := \{z_1^{(k)}, \ldots, z_n^{(k)}\}$ be sets of new* distinct *extension variables. The CNF formula $\neg\mathrm{INJ}_k$ consists of the following set of clauses:*

1. *$||C^{(k)}(\overline{W}_1)||(\overline{Y}_k)$; $\quad ||C^{(k)}(\overline{W}_2)||(\overline{Z}_k)$ (expresses that $\overline{Y}_k, \overline{Z}_k$ are the output bits of $C^{(k)}(\overline{W}_1), C^{(k)}(\overline{W}_2)$, respectively);*
2. *$\neg u_i \vee \neg y_i^{(k)} \vee z_i^{(k)}$; $\quad \neg u_i \vee y_i^{(k)} \vee \neg z_i^{(k)}$, for all $1 \leq i \leq n$ (expresses that u_i implies $y_i^{(k)} \equiv z_i^{(k)}$);*
3. *$v_i \vee w_i^{(1)} \vee w_i^{(2)}$; $v_i \vee \neg w_i^{(1)} \vee \neg w_i^{(2)}$; for al $1 \leq i \leq m$ (expresses that $\neg v_i$ implies $w_i^{(1)} \equiv w_i^{(2)}$);*
4. *$u_1, \ldots, u_n$ (expresses that $\overline{Y} \equiv \overline{Z}$);*
5. *$\neg v_1 \vee \ldots \vee \neg v_m$ (expresses that $\overline{W}_1 \not\equiv \overline{W}_2$);*

For simplicity of writing we introduce the following notation: Let ℓ be a literal and let A be a CNF formula. We denote by $\ell \circledvee A$ the set of clauses (that is, the CNF formula) that results by adding to each clause of A the literal ℓ.

We now encode as a CNF formula denoted by $\neg$INJ the negation of

$$\bigwedge_{k=1}^{t} \big(C^{(k)}(\overline{W}_1) \equiv C^{(k)}(\overline{W}_2) \ \rightarrow \overline{W}_1 \equiv \overline{W}_2\big).$$

Definition 7.5.3 ($\neg$INJ) *The* CNF *formula* $\neg$INJ *consists of the following set of clauses:*

[7]We say that a Boolean function f does not depend on an input bit w_i if for all input assignments α to f, flipping the truth value of w_i in α does not change the value of f.

1. *$\neg p_k \circledvee \neg \mathrm{INJ}_k$ for al $1 \leq k \leq t$ (expresses that INJ$_k$ implies $\neg p_k$);*
2. *$p_1 \vee \ldots \vee p_t$ (expresses $\vee_{k=1}^{t} \neg \mathrm{INJ}_k$.)*

In a similar manner one can encode as a CNF the negation of the formula

$$\bigwedge_{1 \leq i < j \leq t} \big(C^{(i)}(\overline{W}_1) \not\equiv C^{(j)}(\overline{W}_2)\big),$$

denoted by $\neg$DSJ (where DSJ stands for *disjoint*). We shall not develop the encoding precisely, as this is pretty much similar to $\neg$INJ.

The last part of the promise axiom we need to encode is the formula

$$\bigvee_{i=1}^{t} C^{(i)}(\overline{W}_1) \equiv X.$$

We denote the CNF encoding of this formula by RST (which stands for *restriction*). Again, this is similar to the encoding of $\neg$INJ, but we show how to encode it anyway, since we would like to illustrate in the sequel how resolution can use RST to efficiently prove some basic facts about the X variables (in the case the circuits in C have certain simple form).

Definition 7.5.4 (RST) *For every $1 \leq k \leq t$, recall that $\overline{Y}_k := \{y_1^{(k)}, \ldots, y_n^{(k)}\}$ are the output variables of $||C^{(k)}(\overline{W}_1)||$ from Definition 7.5.2. The* CNF *formula* RST *consists of the following set of clauses:*

1. *$\neg f_i^{(k)} \vee \neg y_i^{(k)} \vee x_i$; $\neg f_i^{(k)} \vee y_i^{(k)} \vee \neg x_i$ for all $1 \leq i \leq n$ (expresses that $f_i^{(k)}$ implies $y_i^{(k)} \equiv x_i$);*
2. *$\neg h_k \vee f_1^{(k)}, \ldots, \neg h_k \vee f_n^{(k)}$ (expresses that h_k implies $\overline{Y}_k \equiv X$);*
3. *$h_1 \vee \ldots \vee h_t$* (expresses *$\bigvee_{i=1}^{t} \overline{Y}_k \equiv X$.)*

Finally, the promise axiom PRM$_{C,\Lambda}$ is the following CNF formula:

Definition 7.5.5 (CNF encoding of PRM$_{C,\Lambda}$) *The CNF encoding of the promise axiom PRM$_{C,\Lambda}$ consists of the following clauses:*

1. *$\neg q_1 \circledvee \neg$INJ (expresses that INJ implies $\neg q_1$);*
2. *$\neg q_2 \circledvee \neg$DSJ (expresses that DSJ implies $\neg q_2$);*
3. *$q_1 \circledvee (q_2 \circledvee \mathrm{RST})$ (expresses $\neg\mathrm{INJ} \vee \neg\mathrm{DSJ} \vee \mathrm{RST}$, which is equivalent to $\mathrm{INJ} \wedge \mathrm{DSJ}) \to \mathrm{RST}$).*

7.5.1.3 Proving Basic Facts About Encoded Circuits inside Resolution

The following simple claim illustrates how one can reason inside resolution, and specifically can "eliminate implications" inside resolution. Consider, for instance, line 1 in $\mathrm{PRM}_{C,\Lambda}$ (Definition 7.5.5). This line expresses that INJ implies $\neg q_1$. In other words, it is logically equivalent to $\mathrm{INJ} \to \neg q_1$. Assume that we already know INJ (which formally means that we have a resolution refutation of $\neg$INJ). We would like to arrive inside resolution at $\neg q_1$. The following straightforward claim illustrates how to do this in resolution.

Claim: Let A be an unsatisfiable CNF formula with a resolution refutation of size s and let ℓ be any literal. Then there is a resolution proof of ℓ from $\ell \circledvee A$ of size s.

Proof of claim: Assume that the resolution refutation of A is the sequence of clauses $A_1, \ldots, A_s$, where $A_s = \Box$ (the empty clause). Then $\ell \vee A_1, \ldots, \ell \vee A_s$ is a resolution proof of $\ell \vee \Box = \ell$ from $\ell \circledvee A$ (we assume that ℓ is not in any A_i; or else, by the weakening rule, the claim also holds).$\blacksquare_{\text{Claim}}$

Note that Claim 7.5.1.3 implies that if there is a refutation of $\neg$INJ of size s, then there is also a proof of $\neg q_1$ of the same size s, from line 1 in $\mathrm{PRM}_{C,\Lambda}$ (Definition 7.5.5).

We now illustrate how resolution can efficiently prove a certain simple fact about simple circuits. This is needed (among other efficient proofs of similar simple facts) in order to show the upper bound in Section 7.3 (and specifically, it is used in Claim 7.3). Other similar facts about the Boolean circuits constructed in Section 7.3 can be proved inside resolution in a similar manner, and we shall not describe these proofs here.

For some $1 \leq k \leq t$, let $C^{(k)}$ be a circuit from a sequence of circuits C (as in the promise axioms), where m and n are the number of input and output variables of $C^{(k)}$, respectively. Assume that the ith output bit of $C^{(k)}$ computes the jth input bit w_j for some $1 \leq j \leq m$ and $1 \leq i \leq n$. We require that resolution can efficiently refute (the encoding via Definition 7.5.2 of):

$$C^{(k)}(\overline{W}_1) \equiv C^{(k)}(\overline{W}_2) \wedge w_j^{(1)} \not\equiv w_j^{(2)}$$

(note that by assumption this is clearly a contradiction).

Since $C_i^{(k)}$ just computes the jth input bit w_j, then in fact we can assume that the encoding $\|C_i^{(k)}(\overline{W})\|(y_i)$ consists of only the single clause w_j (remember that by Definition 7.5.1 we identify between the variable *encoding an input gate* with the input variable w_j itself; and here we know that w_j is also the output variable). Thus, by 2 in Definition 7.5.2 we have that the output bit y_j of $C_i^{(k)}(\overline{W}_1)$ equals the output bit z_j of $C_i^{(k)}(\overline{W}_2)$, and y_j is actually $w_j^{(1)}$ and z_j is actually $w_j^{(2)}$. Therefore, by Definition 7.5.2 3, we can prove v_j. So, by one resolution rule applied to 7.5.2 5, we are left with $\bigvee_{i \neq j} v_i$.

Assume that all but a constant number of the output bits of $C^{(k)}$ compute some (distinct) input bit w_j, for some $1 \leq j \leq m$ (this assumption corresponds to the circuits we build in Section 7.3). Then the process described in the previous paragraphs can be iterated for all such output bits of $C^{(k)}$, in order to cut off (that is, resolve over) all the v_j variables in clause 5 in Definition 7.5.2, until we reach only a disjunction of *constant* *number* of variables v_j instead of clause 5 in 7.5.2.

We are thus left with a *constant number* of circuits depending only on a *constant number* of input variables. Therefore, we can now refute with a polynomial-size resolution refutation the encoding of

$$C^{(k)}(\overline{W}_1) \equiv C^{(k)}(\overline{W}_2) \ \wedge \overline{W}_1 \not\equiv \overline{W}_2 \tag{7.11}$$

(if indeed the circuit $C^{(k)}$ computes an injective map, which means that (7.11) is unsatisfiable).

7.5.1.4 Comments on Decoding the Encoded Promise Axioms

To assert that promise resolution is a Cook-Reckhow proof system (see the first paragraph in Section 1.2.4.1 for a definition) we need to make sure that a promise resolution refutation can be identified as such in polynomial-time. For this, one needs to be able to verify whether a given CNF is an instance of the promise axiom.

As mentioned in Section 7.2, this can be done by "decoding" the CNF that encodes the promise axiom $\text{PRM}_{C,\Lambda}$ and then checking that each circuit in C has the right number of input and output bits. Here we illustrate how this can be achieved.

First, it is possible to identify which are the clauses pertaining to the promise axioms out of all the clauses in the refutation (for instance, any clause used as an axiom that is not one of the clauses of the CNF meant to be refuted). Second, it is possible to identify which are the clauses of the promise axiom that are part of the circuit encoding (that is, clauses in line 1 in Definition 7.5.2). It is then possible to decode the circuits from the encoding, and check that the circuits are legitimate ones and have the intended number of input and output variables (we omit the details).

7.6 Chapter Summary

This chapter establishes a new framework of propositional proof systems that are able to separate the unsatisfiable CNF formulas from the set of CNF formulas having many satisfying assignments. We analyzed the complexity of basic cases pertaining to such proof systems. In particular, we demonstrated an exponential separation between the case of a big promise (a constant fraction of all truth assignments) and the average-case proof complexity of refutations under a smaller promise (that is, a promise of $2^{\delta n}$, for any constant $0 < \delta < 1$).

Chapter 8

Conclusions and Open Problems

In this thesis we explored new proof systems for establishing propositional tautologies and related languages. We studied the complexity of these proof systems and compared their strength to other previously studied proof systems.

In the first parts of the thesis (Chapters 3–5) we focused on semantic algebraic proof systems operating with multilinear formulas and their relations with extensions of the resolution proof system operating with disjunctions of linear equations. Specifically, we showed that multilinear proofs operating with depth-3 multilinear formulas are somewhat close to a certain (proper) fragment of resolution operating with linear equations (that is, the R^0(lin) proof system). Moreover, the monotone interpolation via communication game technique yields exponential-size lower bounds on R^0(lin) proofs, by the results in Chapter 4. Therefore, an important question that remains open is this:

Open problem 1: *Can any super-polynomial lower bound on the size of multilinear proofs operating with depth-*3 *multilinear formulas be proved via the monotone interpolation technique?*

We observed in Chapter 4 that there are polynomial-size R(lin) refutations of the clique-coloring formulas (for certain weak parameters, that is, when the formulas express, loosely speaking, the separation of k-cliques from complete k'-partite graphs, where $k = \sqrt{n}$ and $k' = (\log n)^2/8 \log\log n$). We were unable to polynomially-simulate these R(lin)-refutations by multilinear proofs (of any depth). Thus, the following question (which stands in contrast to *open problem* 1, above) arises:

Open problem 2: *Are there polynomial-size multilinear proofs of the clique-coloring formulas (for these values of k, k', or for stronger values*[1]*).*

A positive answer to this question would show that multilinear proofs do not posses the feasible monotone interpolation property (Definition 4.5.8).[2] Showing polynomial-size multilinear proofs of the clique-coloring formulas would also separate multilinear proofs from cutting planes proofs, by known exponential-size lower bounds on cutting planes proofs (cf. Pudlák (1997)).

With respect to the strength of full R(lin) proofs, the following problem remains open:

Open problem 3: *Demonstrate super-polynomial lower bounds on R(lin) proofs.*

We observed that R(lin) does not possess the monotone interpolation property (it admits polynomial-size proofs of the clique-coloring formulas, as noted above). Further-

[1]That is, for values of k, k' that yield stronger lower-bounds on monotone circuits separating k-cliques from complete k'-partite graphs.

[2]Note that this does not rule out entirely the possibility of using the monotone interpolation technique to obtain exponential-size lower bounds for multilinear proof systems. This is because, if the transformation from refutations of formulas to monotone circuits computing the corresponding interpolant functions, is done with only a *quasi-polynomial* increase in size, then one could still obtain an exponential-size lower bound on the refutations (as the lower bounds on monotone circuits separating k-cliques from complete k'-partite graphs are *exponential* [for the appropriate values of k, k']).

more, it can be shown that R(lin) polynomially simulates Res(k) for any k (for a definition of Res(k) see Krajíček (2001)), and so R(lin) is a considerably strong system (note that some lower bounds for Res(k) proofs use "counting principles" as hard formulas [cf. Atserias et al. (2002), Segerlind et al. (2002), Razborov (2002-2003)]; while R(lin) [and even its proper subsystem R^0(lin)] is suited to efficiently prove these formulas). Therefore, providing super-polynomial lower bounds on R(lin) proofs would probably require new techniques to be developed.

Another question that arises with respect to the R(lin) system concerns the relative strength gained by proofs operating with *equalities* versus *inequalities*. In this respect, two questions seem relevant:

Open problem 4: *Does the cutting planes proof system (either with polynomially bounded coefficients or not) polynomially simulate* R^0(lin)*?*

Open problem 5: *Does* R(CP*) *polynomially simulate* R(lin)*?* (We have shown that the converse simulation is true.)

The following problem is most probably easy to accomplish:
Open problem 6: *Can the efficient refutations of the pigeonhole principle and the Tseitin mod p formulas in* R^0(lin) *be transformed directly into proofs in algebraic proof systems of a Cook-Reckhow type that operate with general (namely, not necessarily multilinear) arithmetic formulas?* This in turn would simplify corresponding algebraic proofs (over the field of rationals) for the corresponding two (families of) formulas demonstrated in Grigoriev and Hirsch (2003).

After investigating semantic algebraic propositional proof systems, we turned into symbolic proofs that establish polynomial identities written as arithmetic formulas. Such symbolic proofs were already considered in the literature on algebraic propositional proof systems (cf. Buss et al. (1996/97); Grigoriev and Hirsch (2003)) (see the discussion in Section 1.2.3.2): in this setting one usually requires that any proof-line (that is, an arithmetic formula) in the algebraic propositional proof is treated as a ("syntactic") term and is derived as such from previous proof-lines via the polynomial-ring axioms applied on any subformula. (This way, one obtains a Cook-Reckhow proof system operating with arithmetic formulas, instead of semantic algebraic proof systems.)

Our study was thus motivated, among other things, by the question whether semantic algebraic proof systems are stronger than their Cook-Reckhow counterparts. We demonstrated an exponential lower bound on proofs in an analytic fragment of full symbolic proof systems, operating with depth-3 formulas under a structural regularity condition. The main general problem left open is of course the following:

Open problem 7: *Improve the lower bound result to stronger fragments of symbolic proofs of polynomial identities, than that shown in Chapter 6.*

In particular it would be interesting to see whether there exist short general (that is, non-analytic) depth-3 symbolic proofs for the identities based on the symmetric polynomials. In case no such short proofs exist (or even no such short proofs operating with only depth-3 *multilinear* formulas), it would imply that the short (semantic) multilinear proofs demonstrated in this thesis do not transform naturally into syntactic algebraic proofs.

In Chapter 7 we introduced an studied the complexity of a new framework of propositional proof systems that are able to separate the unsatisfiable CNF formulas from the set of CNF formulas having many satisfying assignments. We used resolution as the underlying proof system. One question we have not addressed is the following:

Open problem 8: *What can be gained (if at all) when one augments a stronger proof system than resolution, like bounded-depth Frege proof system or Frege proof system, with the promise axioms (for a small promise like $2^{\delta n}$; since for a big promise we showed that already resolution can efficiently refute all unsatisfiable 3CNF formulas)?*

Another question that arises is this:

Open problem 9: *Does the fact that we require the Boolean circuits in the promise axioms to be* provably injective *and to* provably posses disjoint images *(that is, provably inside resolution) constitutes a real restriction with respect to the sizes of refutations?*

(Note that the lower bound proof for resolution under the promise $2^{\delta n}$ in Section 7.4 did not make use of the requirements that the Boolean circuits in the promise axioms should be provably injective and to provably posses disjoint images.)

In other words, we ask whether there is a sequence of circuits $C^{(1)}, \dots, C^{(t)}$ for which adding the axiom $\vee_{i=1}^{t} C^{(i)}(\overline{W}) \equiv X$ (where the parameter t and the number of variables in m are taken from the smaller promise axiom 7.2.4) to resolution (or a stronger proof system) gives a super-polynomial speed-up for some contradictory family of formulas over standard resolution (or the stronger proof system); but that we cannot prove efficiently in resolution (or the stronger proof system) that $C^{(1)}, \dots, C^{(t)}$ are injective or that they have pairwise disjoint images?

A different and more general task is to come up with other natural models of propositional proof systems that capture a "relaxed" notion of soundness. For instance, Pitassi (2006) suggested considering "approximate proofs" in the framework of algebraic proof systems.

Finally, we have not dealt directly in this chapter with the promise $\Lambda = 2^n/poly(n)$, though a similar upper bound (with a similar proof) to that shown in Section 7.3 might also holds for this promise (when the promise axiom is modified accordingly). In this respect it is worth mentioning that Krajíček (2007) observed that the work of Razborov and Rudich (1997) implies the existence (under a cryptographic conjecture) of a Boolean function g with n^δ input bits (denoted by $y_1, \dots, y_{n^\delta}$) and n output bits (denoted by $g_1(y_1, \dots, y_{n^\delta}), \dots, g_n(y_1, \dots, y_{n^\delta})$), for any constant $0 < \delta < 1$, that has the following property: given any CNF formula K in n variables $x_1, \dots, x_n$, substituting $g_1(y_1, \dots, y_{n^\delta}), \dots, g_n(y_1, \dots, y_{n^\delta})$ for the original x_i variables in K yields a new CNF formula that is unsatisfiable only if K has at most $2^n/n^{\omega(1)}$ satisfying assignments. This means that *under the promise $2^n/poly(n)$ the substitution g is sound*: any unsatisfiable CNF formula (clearly) stays unsatisfiable after the substitution, while any CNF with more than $2^n/poly(n)$ satisfying assignments stays satisfiable after the substitution.

Bibliography

Scott Aaronson (2004). Multilinear formulas and skepticism of quantum computing. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 118–127, Chicago, IL, 2004. ACM Press. 1.2.1.1

Michael Alekhnovich and Alexander A. Razborov (2001). Lower bounds for polynomial calculus: non-binomial case. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 190–199. IEEE Computer Soc., Los Alamitos, CA, 2001. 1.2.1.1

Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson (2002). Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002. 1.1.2, 2.4.0.2, 3.5

Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson (2004). Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. 1.2.1.1, 1.2.1.2, 7.5.1.1

Noga Alon (1986). Eigenvalues and expanders. *Combinatorica*, 6:83–96, 1986. 3.5

Noga Alon and R. Boppana (1987). The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987. 4.6, 4.6.2

A. E. Andreev (1985). On a method for obtaining lower bounds for the complexity of individual monotone functions. *Dokl. Akad. Nauk SSSR (in Russian)*, 282(5):1033–1037, 1985. [Engl. Transl. Soviet Math. Dokl., vol. 31 (1985), pp. 530-534]. 4.6

A. Atserias, Maria Luisa Bonet, and J. Esteban (2002). Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. *Information and Computation*, 176:152–136, August 2002. 3, 4.4.3, 4.4.3, 4.4.13, 4.4.3, 8

Paul Beame and Toniann Pitassi (1998). Propositional proof complexity: past, present, and future. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, (65):66–89, 1998. 1.1, 1.2.1.1

Paul Beame and Søren M. Riis (1998). More on the relative strength of counting principles. In *Feasible Arithmetic and Proof Complexity*, DIMACS. 1998. 1.1.1

Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák (1996). Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Math. Soc. (3)*, 73(1):1–26, 1996. 1.1.1

Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks (2002). The efficiency of resolution and Davis-Putnam procedures. *SIAM J. Comput.*, 31(4):1048–1075, 2002. 1.1, 1.2.4.1, 8, 7.4.5

Eli Ben-Sasson (2001). *Expansion in Proof Complexity*. PhD thesis, Hebrew University, Jerusalem, Israel, September 2001. 8

Eli Ben-Sasson (2002). Hard examples for the bounded depth Frege proof system. *Computational Complexity*, 11(3-4):109–136, 2002. 1.2.1.2, 3.5.1

Eli Ben-Sasson and Russell Impagliazzo (1999). Random CNF's are hard for the polynomial calculus. In *Proceedings of the IEEE 40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pages 415–421. IEEE Computer Soc., Los Alamitos, CA, 1999. 1.2.1.1, 1.2.1.2

Eli Ben-Sasson and Avi Wigderson (2001). Short proofs are narrow—resolution made simple. *J. ACM*, 48(2):149–169, 2001. 1.2.1.2, 1.2.4.1, 8, 1.2.4.2, 3.5.1, 7.1.1, 7.1.1, 7.4, 7.4.1, 7.4.1, 7.4.1, 7.4.1

Archie Blake (1937). *Canonical expression in boolean Algebra*. PhD thesis, University of Chicago, 1937. 1.1.2

Maria Luisa Bonet, Toniann Pitassi, and Ran Raz (1997). Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997. 1.2.2.1, 1.2.2.2, 4.4.3, 4.4.3, 4.5.1.1, 4.6, 4.6.4

Samuel R. Buss (1986). *Bounded Arithmetic*, volume 3 of *Studies in Proof Theory*. Bibliopolis, 1986. 1.1

Samuel R. Buss (1997). *Bounded arithmetic and propositional proof complexity*, pages 67–121. In: Logic of Computation, H. Schwichtenberg, ed. Springer-Verlag, Berlin, 1997. 1.1

Samuel R. Buss and Toniann Pitassi (1997). Resolution and the weak pigeonhole principle. In *Computer science logic (Aarhus, 1997)*, volume 1414 of *Lecture Notes in Comput. Sci.*, pages 149–156. Springer, Berlin, 1997. 4.4.3

Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiří Sgall (1996/7). Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1996/97. 1.1.1, 1.2.1.1, 1.2.3.2, 8

Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi (2001). Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. System Sci.*, 62(2):267–289, 2001. Special issue on the 14th Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999). 1.2.1.1, 1.2.1.2, 3.5, 3.5.3, 3.5, 3.5, 3.5.6, 4

Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo (1996). Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 174–183, New York, 1996. ACM. 1.1.1, 1.1.2, 2.4.0.1, 3.5

Stephen Cook (2005). *Theories for Complexity Classes and Their Propositional Translations*, pages 175–227. Complexity of computations and proofs, Jan Krajíček, ed. Quaderni di Matematica, 2005. 1.1

Stephen Cook and Phuong Nguyen (2004–2008). *Foundations of Proof Complexity: Bounded Arithmetic and Propositional Translations*. 2004–2008. Book in preparation, available at url: `http://www.cs.toronto.edu/~sacook/`. 1.1

Stephen A. Cook (1971). The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing*, pages 151–158. ACM, New York, 1971. 7.3, 7.3

Stephen A. Cook and Robert A. Reckhow (1979). The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979. 1.1, 1.1

Nachum Dershowitz and Iddo Tzameret (2007). Complexity of propositional proofs under a promise. In *Proceedings of the Thirty-Fourth International Colloquium on Automata, Languages and Programming (ICALP)*, pages 291–302, 2007.

Zeev Dvir and Amir Shpilka (2006). Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006. 1.2.3.1, 3.0.4.1, 3.5.1

Zeev Dvir, Amir Shpilka, and Amir Yehudayoff (2008). Hardness-randomness tradeoffs for bounded depth circuits. In *Proceedings of the 40th Annual ACM Symposium on the Theory of Computing*, pages 741–748, 2008. 1.2.3.1

Uriel Feige, Jeong Han Kim, and Eran Ofek (2006). Witnesses for non-satisability of dense random 3CNF formulas. In *Proceedings of the IEEE 47th Annual Symposium on Foundations of Computer Science*, 2006. 1.2.4.1

Dima Grigoriev and Edward A. Hirsch (2003). Algebraic proof systems over formulas. *Theoret. Comput. Sci.*, 303(1):83–102, 2003. Logic and complexity in computer science (Créteil, 2001). 1.2.1.1, 1.2.3.2, 3.5, 5.3.0.1, 8

Armin Haken (1985). The intractability of resolution. *Theoret. Comput. Sci.*, 39(2-3): 297–308, 1985. 1.2.1.2, 5.3.0.1

Edward Hirsch (1998). A fast deterministic algorithm for formulas that have many satisfying assignments. *Logic Journal of the IGPL*, 6(1):5971, 1998. 1.2.4.1, 5

Edward Hirsch and Arist Kojevnikov (2006). Several notes on the power of Gomory-Chvátal cuts. *Ann. Pure Appl. Logic*, 141:429–436, 2006. 1.2.2.1

Edward Hirsch, Dmitry Itsykson, Arist Kojevnikov, Alexander Kulikov, and Sergey Nikolenko (2005). Report on the mixed boolean-algebraic solver. Technical report, Laboratory of Mathematical Logic of St. Petersburg Department of Steklov Institute of Mathematics, November 2005. url: `http://logic.pdmi.ras.ru/˜basolver/basolver-firstreport.pdf`. 1.1.1, 1.2.2.1

Pavel Hrubeš (2008). Equational calculi. April 2008. Preprint. 4

R. Impagliazzo and N. Segerlind (2001). Counting axioms do not polynomialy simulate counting gates. In *Proceedings of the 42rd IEEE Symposium on Foundations of Computer Science*, pages 200–109. 2001. 1.1.1

R. Impagliazzo and N. Segerlind (2002). Bounded-depth frege systems with counting axioms polynomially simulate nullstellensatz refutations. In *Proceedings of the Twenty-Ninth Annual Colloquium on Automata, Languages and Programming*, pages 208–219. 2002. 1.1.1

Russel Impagliazzo, Toniann Pitassi, and Alasdair Urquhart (1994). Upper and lower bounds for tree-like cutting planes proofs. In *Ninth Annual Symposium on Logic in Computer Science*, pages 220–228. IEEE Comput. Soc. Press, 1994. 4.5.1.1

Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall (1999). Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2): 127–144, 1999. 1.1, 1.2.1.1, 1.2.1.2, 3.5, 5.3.0.1

Valentine Kabanets and Russell Impagliazzo (2004). Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. 1.2.3.1

Mauricio Karchmer and Avi Wigderson (1988). Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 539–550. ACM, 1988. 4.5.1.1, 4.5.2

Zohar Karnin and Amir Shpilka (2008). Deterministic black box polynomial identity testing of depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity (CCC '08)*, pages 280-291, 2008. 1.2.3.1

Neeraj Kayal and Nitin Saxena (2007). Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007. 1.2.3.1, 3.0.4.1, 3.5.1

Arist Kojevnikov (2007). Improved lower bounds for tree-like resolution over linear inequalities. In *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 70–79. Springer, 2007. 1.2.2.1, 4.7

Jan Krajíček (1994). Lower bounds to the size of constant-depth propositional proofs. *The Journal of Symbolic Logic*, 59(1):73–86, 1994. 4.5

Jan Krajíček (1995). *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1995. 1.1, 2.2

Jan Krajíček (1997). Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997. 1.2.2.1, 1.2.2.2, 4.4.3, 4.5.1, 4.5.1.1, 4.5.4, 4.5.5, 4.5.3

Jan Krajíček (1998). Discretely ordered modules as a first-order extension of the cutting planes proof system. *The Journal of Symbolic Logic*, 63(4):1582–1596, 1998. 1.1.2, 1.2.2.1, 1.2.2.2, 4.4.3, 4.7

Jan Krajíček (2001). On the weak pigeonhole principle. *Fund. Math.*, 170(1-2):123–140, 2001. Dedicated to the memory of Jerzy Łoś. 4.4.3, 8

Jan Krajíček (2007). Substitutions into propositional tautologies. *Information Processing Letters*, 101:163–167, 2007. 8

Jan Krajíček (2007). Personal communication, 2007. 7.2.3

Jan Krajíček (2007). An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *The Journal of Symbolic Logic*, 73(1): 227-237, 2008. 4.4.3

Jan Krajíček, Pavel Pudlák, and Alan Woods (1995). An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures Algorithms*, 7(1):15–39, 1995. 1.2.1.2, 5.3.0.1

Toniann Pitassi (1997). Algebraic propositional proof systems. In *Descriptive complexity and finite models (Princeton, NJ, 1996)*, volume 31 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 215–244. Amer. Math. Soc., Providence, RI, 1997. 1.1.1, 1.2.1.1, 3.0.4.1

Toniann Pitassi (2006). Using hardness in proof complexity, April 2006. Talk given in *New Directions in Proof Complexity*, an Isaac Newton institute workshop, Cambridge. 8

Toniann Pitassi, Paul Beame, and Russell Impagliazzo (1993). Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993. 1.2.1.2, 5.3.0.1

Pavel Pudlák (1997). Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997. 4.4.3, 10, 8

Ran Raz (2004). Multi-linear formulas for permanent and determinant are of super-polynomial size. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing*, pages 633–641, Chicago, IL, 2004. ACM. 1.2.1.1

Ran Raz (2006). Separation of multilinear circuit and formula size. *Theory of Computing, Vol. 2, article 6*, 2006. 1.2.1.1

Ran Raz and Amir Shpilka (2005). Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005. 1.2.3.1, 3.0.4.1

Ran Raz and Iddo Tzameret (2008). The strength of multilinear proofs. *Computational Complexity*, 17(3):407–457, 2008.

Ran Raz and Iddo Tzameret (2008). Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

Alexander A. Razborov (1985). Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR (in Russian)*, 281(4):798–801, 1985. [English translation in Sov. Math. Dokl., vol . 31 (1985), pp. 354-357.]. 4.6

Alexander A. Razborov (2003). Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Manuscript*, 2002-2003. 1.2.1.1, 8

Alexander A. Razborov (1995). Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izv. Ross. Akad. Nauk Ser. Mat.*, 59(1):201–224, 1995. 4.5.1.1

Alexander A. Razborov (1996). Lower bounds for propositional proofs and independence results in bounded arithmetic. In *Automata, languages and programming (Paderborn, 1996)*, volume 1099 of *Lecture Notes in Comput. Sci.*, pages 48–62. Springer, Berlin, 1996. 1.1

Alexander A. Razborov (1998). Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998. 1.2.1.1, 1.2.1.2, 5.3.0.1

Alexander A. Razborov and Steven Rudich (1997). Natural proofs. *J. Comput. System Sci.*, 55(1, part 1):24–35, 1997. 8

J. Alan Robinson (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):2341, 1965. 1.1.2

Jacob T. Schwartz (1980). Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. 1.2.3.1, 3.0.4.1

Nathan Segerlind (2007a). Nearly-exponential size lower bounds for symbolic quantifier elimination algorithms and OBDD-based proofs of unsatisfiability. *Electronic Colloquium on Computational Complexity*, January 2007a. ECCC, TR07-009. 1.1

Nathan Segerlind (2007b). The complexity of propositional proofs. *Bull. Symbolic Logic*, 13(4):417–481, 2007b. 1.1

Nathan Segerlind, Sam Buss, and Russell Impagliazzo (2002). A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM J. Comput.*, 33(5): 1171–1200, 2004. 8

Amir Shpilka and Ilya Volkovich (2008). Read-once polynomial identity testing. In *Proceedings of the 40th Annual ACM Symposium on the Theory of Computing*, pages 507–516, 2008. ACM. 1.2.3.1

Amir Shpilka and Avi Wigderson (2001). Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10:1–27, 2001. 2.5.4

Luca Trevisan (2004). A note on approximate counting for k-DNF. In *Proc. 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2004)*, Lecture Notes in Computer Science, vol. 3122, Springer, pp. 417–426, 2004. 1.2.4.1, 1.2.4.2, 7.3

Grigori Tseitin (1968). *On the complexity of derivations in propositional calculus*. Studies in constructive mathematics and mathematical logic Part II. Consultants Bureau, New-York-London, 1968. 3.5

Iddo Tzameret (2008). *On the Structure and Complexity of Symbolic Proofs of Polynomial Identities*. Manuscript, 35 pages, April 2008.

Alasdair Urquhart (1987). Hard examples for resolution. *Journal of the ACM*, 34(1): 209–219, 1987. 1.2.1.2, 3.5.1

Heribert Vollmer (1999). *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999. 9

Richard Zippel (1979). Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*. Springer-Verlag, 1979. 1.2.3.1, 3.0.4.1

המערכות בפרק 7 להיות מערכות *הפרכה* (כלומר, כאלו אשר מוכיחות את האי-ספיקות של נוסחאות CNF).

בהתאם לכך, נגדיר את המערכת *רזולוציה תחת הבטחה* Λ, עבור Λ פונקציה של n מספר המשתנים של ה-CNF, להיות מערכת הפרכה פסוקית המרחיבה את רזולוציה, בה לכל נוסחת CNF לא ספיקה יש הפרכה במערכת ואילו לכל נוסחת CNF בעלת יותר מ-Λ השמות מספקות אין הפרכה במערכת. בצורה זו אנו מקבלים מערכת הוכחה שהיא שלמה ונאותה לקבוצת נוסחאות ה-CNF תחת ההבטחה שאם נוסחה אינה ספיקה הרי שיש לה יותר מ-Λ השמות לא מספקות.

אנו מראים ששינוי גודל ההבטחה Λ מוביל להפרדה אקספוננציאלית (במקרה הממוצע) בין מערכות הוכחה בעלות הבטחות Λ גדולות יותר לאלו עם הבטחות Λ קטנות יותר, כלהלן.

חסם עליון:

יהא $0<\varepsilon<1$ קבוע כלשהו ויהא $\Lambda=\varepsilon\cdot 2^n$ פרמטר ההבטחה הנתון (קרי, שבר קבוע מסך כל ההשמות האפשריות). אזי לכל 3CNF עם n משתנים לא ספיק יש הפרכת רזולוציה בגודל פולינומיאלי ב-n תחת ההבטחה Λ.

חסם תחתון:

יהא $0<\delta<1$ קבוע כלשהו ויהא $\Lambda = 2^{\delta n}$ פרמטר ההבטחה הנתון. אזי ל-3CNF *מקרי* עם n משתנים אין הוכחת רזולוציה תת-אקספוננציאלית (ב-n) תחת ההבטחה Λ, בהינתן שמספר הפסוקיות ב-3CNF הוא $O(n^{3/2-\varepsilon})$, עבור $0<\varepsilon<½$.

התוצאות אודות מערכות הוכחה תחת הבטחה פורסמו ב:

Nachum Dershowitz and Iddo Tzameret. **Complexity of propositional proofs under a promise**. *Proceedings of the Thirty-Fourth International Colloquium on Automata, Languages and Programming (ICALP)*, pages 291–302, 2007.

שימושים להוכחות מולטי-לינאריות:

פרק 5 מוקדש להוכחת המשפט הבא, המקשר בין $R^0(\text{lin})$ לבין הוכחות מולטי-לינאריות:

משפט: הוכחות מולטי-לינאריות בהן כל שורה היא נוסחה מולטי-לינארית מעומק 3 מעל שדה עם מציין 0, מסמלצות פולינומיאלית הוכחות ב-$R^0(\text{lin})$.

מסקנה מיידית של משפט זה והחסמים העליונים על הוכחות $R^0(\text{lin})$ המובאים בפרק 4, הם החסמים העליונים על הוכחות מולטי-לינאריות של עקרון שובך היונים וטאוטולוגיות צייטין (Tseitin) המבוססות על גרפים.

התוצאות אודות רזולוציה מעל דיסיונקציות של שויונות לינאריים ושימושים להוכחות מולטי-לינאריות פורסמו ב:

Ran Raz and Iddo Tzameret. **Resolution over linear equations and multilinear proofs**. *Annals of Pure and Applied Logic,* 155(3):194-224, 2008.

הוכחות סימבוליות של שויונות פולינומיים

בפרק 6 נציג מערכות הוכחה בהן כל הוכחה מהווה עדות לכך שנוסחה אריתמטית נתונה מחשבת את פולינום האפס. כללי היסק של מערכות אלו יגזרו מהאקסיומות של חוג-הפולינומים. בפרט, תהי Φ נוסחה אריתמטית מעל שדה F כלשהו המחשבת את פולינום האפס. הוכחה סימבולית לכך ש-Φ מחשבת את פולינום האפס היא סדרה של נוסחאות אריתמטיות, המתחילה בנוסחה Φ ומסתיימת בנוסחה 0, כך שכל שורה בהוכחה (פרט לשורה הראשונה) נובעת מהשורה הקודמת לה על-ידי החלפה של תת-נוסחה בתת-נוסחה אחרת בהתאם לאקסיומות של חוג-הפולינומים מעל השדה F.

באופן אנאלוגי למושג של הוכחות אנליטיות במערכות פסוקיות בוליאניות, נגדיר מערכת הוכחה סימבולית אנליטית לשויונות פולינומיים: הוכחה סימבולית תקרא אנליטית אם לאורך ההוכחה לא ניתן להכניס נוסחאות (או תת-נוסחאות) שאינן נובעות מהנוסחה התחילית. לדוגמה, בהינתן הנוסחה התחילית Φ, לא ניתן להסיק ממנה את הנוסחה Φ+f-f, עבור נוסחה f כלשהי.

בפרק 6 סעיף 6.5 נוכיח חסם תחתון - הראשון מסוגו - על הוכחות אנליטיות של שויונות פולינומיים המשתמשות בנוסחאות אריתמטיות מעומק 3, תחת תנאי מבני מסויים (הדומה לתנאי המגביל את ההוכחות להיות דמויות-עץ). השויונות הפולינומיים עבורם נוכיח את החסם התחתון מבוססים על נוסחאות קטנות מעומק 3 לפולינומים הסימטריים המולטי-לינאריים בהם השתמשנו בפרקים הקודמים.

חסמים על הוכחות פסוקיות תחת הבטחה

בפרק 7 נציג מערכות הוכחה פסוקיות (כהרחבה אפשרית של כל מערכת המכילה את רזולוציה) שהן שלמות ונאותות עבור קבוצת הטאוטולוגיות הפסוקיות, תחת ההבטחה שאם נוסחה אינה ספיקה הרי שיש לה "הרבה" השמות לא מספקות. כיוון שנעסוק בעיקר במערכת הרזולוציה והרחבות שלה, נגדיר את

מערכת $R(lin)$: זו המערכת החזקה יותר בה כל שורת-הוכחה נכתבת כדיסיונקציה של משוואות לינאריות.

מערכת $R^0(lin)$: זו המערכת המהווה תת-מערכת ממש של $R(lin)$ לעיל. כל שורת-הוכחה ב-$R^0(lin)$ היא דיסיונקציה של משוואות לינאריים עם מקדמים קבועים למשתנים (אך לא בהכרח למקדם החופשי, קרי, לקבוע במשוואה), תחת ההגבלה שהדיסיונקציה ניתנת לחלוקה למספר קבוע של תת-דיסיונקציות כך שכל תת-דיסיונקציה מכילה את אותה הצורה הלינארית (פרט לקבוע החופשי) או שהתת-דיסיונקציה היא (תרגום של) פסוקית.

חסמים עליונים:
בפרק 4 נדגים חסמים עליונים על הוכחות $R^0(lin)$ ו-$R(lin)$ כלהלן:

1. הוכחות פולינומיאליות של עקרון שובך היונים הפסוקי ב-$R^0(lin)$;
2. הוכחות פולינומיאליות של טאוטולוגיות צייטין (Tseitin) המבוססות על גרפים ב-$R^0(lin)$;
3. הוכחות פולינומיאליות של טאוטולוגיות המבוססות על עקרון הקליק-צביעה (clique-coloring) ב-$R(lin)$.

תוצאות אינטרפולציה:
בפרק 4 סעיף 4.5 נציג חסמים עליונים פולינומיאליים על אינטרפולנטים (interpolants) המתאימים להוכחות $R^0(lin)$; כלומר, נראה שכל הוכחת $R^0(lin)$ של נוסחה נתונה כלשהי ניתן להמיר למעגל בוליאני (לא בהכרח מונוטוני) בעל גודל פולינומיאלי (בהוכחת ה-$R^0(lin)$ הנתונה) המחשב את פונקציה האינטרפולציה של הנוסחה (אם קיימת כזו פונקציה). נשתמש במשפט האינטרפולציה למערכות הוכחה סמנטיות של Krajíček [1997].

חסמים תחתונים:
בפרק 4 סעיף 4.5 נציג את החסם התחתון הבא:
משפט: *אין הוכחות תת-אקספוננציאליות לטאוטולוגיות המבוססות על עקרון הקליק-צביעה (clique-coloring) ב-$R^0(lin)$.*

משפט זה יוכח על-ידי שימוש בתוצאה של Bonet, Pitassi & Raz (Bonet et al. [1997]), שהיא (בצורה מובלעת) תוצאת אינטרפולציה.

קשרים עם הרחבות של מערכת ה-cutting planes:
Krajíček [1998] הציג מערכת פסוקית המשלבת בין רזולוציה למערכת ה-cutting planes. המערכת סומנה ב-$R(CP)$ (ראה סעיף 4.6 להגדרה). כאשר המקדמים באי-שויונים במערכת $R(CP)$ חסומים פולינומיאלית המערכת המתקבלת מסומנת על-ידי $R(CP^*)$. בסעיף 4.6 אנו מראים את תוצאת הסימולציה הבאה:

משפט: *$R(lin)$ מסמלצת פולינומיאלית את $R(CP^*)$.*

איננו יודעים אם הטענה ההפוכה מתקיימת (כלומר, אם $R(CP^*)$ מסמלצת פולינומיאלית את $R(lin)$).

אנו מציגים תוצאת סימולציה כללית המאפשרת לעבור מהוכחת PCR להוכחה מולטי-לינארית על-ידי הפיכת כל שורת-הוכחה ב-PCR לפולינום מולטי-לינארי, במידה וקיימת נוסחה מולטי-לינארית קטנה לפולינום המולטי-לינארי המתאים לשורת-ההוכחה.

חסמים עליונים על הוכחות מולטי-לינאריות:
בפרק 3 סעיף 3.5, נראה חסמים עליונים פולינומיאליים על הוכחות מולטי-לינאריות המשתמשות בנוסחאות מעומק 3 של טאוטולוגיות צייטין (Tseitin) המבוססות על גרפים.
בפרק 5, נוכיח עוד חסמים עליונים על הוכחות מולטי-לינאריות (נעשה זאת בעזרת תוצאת סימולצית כללית [מסקנה 5.2.5] המבוססת על תוצאות מפרק 4):

- הוכחות מולטי-לינאריות המשתמשות בנוסחאות מעומק 3 של עקרון שובך היונים הפסוקי;
- הוכחות מולטי-לינאריות המשתמשות בנוסחאות מעומק 3 של טאוטולוגיות צייטין (Tseitin) המבוססות על גרפים.

מכך ומתוצאות קודמות אנו מקבלים הפרדה אקספוננציאלית בין הוכחות מולטי-לינאריות מעומק 3 לבין מערכות ההוכחה רזולוציה, PC ו-PCR.

קשר עם חסמים תחתונים על מעגלים אריתמטיים:
בפרק 3 סעיף 3.4, אנו משתמשים בתוצאת הסימולציה הכללית שנידונה לעיל כדי להוכיח את הטענה הבאה: הפרדה סופר-פולינומיאלית מפורשת של הוכחות אלגבריות המשתמשות במעגלים אריתמטיים (כלליים) לבין הוכחות אלגבריות המשתמשות במעגלים אריתמטיים מולטי-לינאריים, גוררת הפרדה סופר-פולינומיאלית בין מעגלים אריתמטיים (כלליים) לבין מעגלים אריתמטיים מולטי-לינאריים.

חלק מהחסמים העליונים להוכחות מולטי-לינאריות נובעים מתוצאות אודות הרחבות של רזולוציה, בהן כל שורת-הוכחה היא דיסיונקציה של שויונות לינאריים עם מקדמים שלמים, כפי שנראה בסעיף הבא.

התוצאות אודות הוכחות מולטי-לינאריות פורסמו כחלק מהמאמרים הבאים:

Ran Raz and Iddo Tzameret. **The strength of multilinear proofs**. *Comput. Complexity ,* 17(3):407-457, 2008.

Ran Raz and Iddo Tzameret. **Resolution over linear equations and multilinear proofs**. *Annals of Pure and Applied Logic,* 155(3):194-224, 2008.

רזולוציה מעל דיסיונקציות של שויונות לינאריים ושימושים להוכחות מולטי-לינאריות

בפרק 4 נגדיר שתי מערכות פסוקיות, בעלות כח שונה, המשלבות היסקים אלגבריים ולוגיים כהרחבות של מערכת הרזולוציה. כל שורת-הוכחה במערכות אלו היא דיסיונקציה של משוואות לינאריות עם מקדמים שלמים הנכתבים בייצוג אונרי. *הגודל* של משוואה לינארית $a_1x_1 + \cdots + a_nx_n = a_0$ מוגדר להיות סכום הערכים המוחלטים של המקדמים $a_0, \dots, a_n$ (כאשר המקדמים נכתבים ביצוג אונרי), כלומר, $\sum_{i=0}^{n} |a_i|$. הגודל של דיסיונקציה של משוואות לינאריות הוא סכום הגדלים של כל המשוואות הלינאריות בדיסיונקציה.

מערכות הוכחה פסוקיות קונקרטיות

בסעיף זה נסקור בקצרה שתי משפחות ספציפיות של מערכות הוכחה. המשפחה הראשונה מכילה מערכות הוכחה לוגיות סטנדרטיות, בהן כל שורת-הוכחה נכתבת כנוסחה בוליאנית (עם הקשרים הלוגיים *וגם*, *או* ו*שלילה*). מערכות אלו נקראות *מערכות פְּרֶגֶה* (או *מערכות מסוג הילברט*). במערכת פרגה נתונה יש קבוצה סופית של (סכימות של) אקסיומות וכללי היסק נאותים. הוכחת-פרגה של טאוטולוגיה τ היא סדרה של נוסחאות בוליאניות המסתיימת בנוסחה τ, כך שכל שורת-הוכחה היא או אקסיומה או נובעת משורות קודמות על-ידי הפעלה של כלל היסק כלשהו. מערכות פרגה הן מערכות שלמות ונאותות, במובן שנוסחה היא טאוטולוגיה אם ורק אם קיימת לה הוכחה במערכת פרגה הנתונה.

משפחת מערכות ההוכחה השניה אותה נזכיר משחקת תפקיד מרכזי בתיזה. זוהי משפחת מערכות ההוכחה האלגבריות הפסוקיות. מערכות אלו, מטרתן, כמקודם, להוכיח טאוטולוגיות פסוקיות. אולם בניגוד למערכות פרגה, כל שורה בהוכחה אלגברית היא *פולינום רב-משתנים מעל שדה* (השדה נקבע מראש). במערכות אלגבריות פסוקיות מוכיחים טאוטולוגיות (או באופן שקול, *מפריכים סתירות*) על-ידי שימוש בכללי היסק אלגבריים בסיסיים. לדוגמה, מהפולינום p ו-q ניתן להסיק את $p+q$ ומהפולינום p בלבד ניתן להסיק את $p\cdot q$. כאשר הסמנטיקה המיועדת של שורת-הוכחה כגון p היא קבוצת הפתרונות של המשוואה $p=0$, ניתן להיווכח שכללי ההיסק האלגבריים שבדוגמה הם אכן נאותים. את השלמות של מערכות הוכחה אלגבריות מוכיחים בדרך-כלל על-ידי שימוש במשפט האפסים של הילברט (Hilbert's Nullstellensatz).

לעיתים ניתן להתייחס למערכות הוכחה אלגבריות כאל מערכות *סמנטיות*. כלומר, מערכות בהן כל שורת-הוכחה נובעת מקודמותיה במובן הסמנטי בלבד, אך לא בהכרח בצורה המפורשת בה נכתבת השורה. במילים אחרות, אין בהכרח אלגוריתם *דטרמיניסטי* יעיל המסוגל להכריע אם שורת ההוכחה אכן נובעת מקודמותיה. למרות זאת, בהסתמך על תוצאות קודמות (Schwartz [1980], Zippel [1979]), ידוע שניתן לבדוק את נכונותן של הוכחות אלגבריות סמנטיות אלו באמצעות אלגוריתם *הסתברותי* יעיל (ראה Pitassi [1997] לסקירה אודות מערכות הוכחה אלגבריות).

בתיזה זו נפתח ונבחן מספר רב של מערכות הוכחה, רובן בעלות אופי אלגברי. התיזה ניתנת לחלוקה לארבעה פרקים מרכזיים כלהלן.

הוכחות מולטי-לינאריות

בפרק 3 אנו מציגים מערכת הוכחה אלגברית פסוקית (סמנטית) בה כל שורת-הוכחה היא פולינום מולטי-לינארי מעל שדה כלשהו (קבוע מראש) הנכתב כנוסחה מולטי-לינארית. לגבי הוכחות מולטי-לינאריות אנו מראים את התוצאות הבאות:

סימולציות:

בפרק 3 אנו מראים כי בעזרת הוכחות מולטי-לינאריות שעובדות עם נוסחאות מולטי-לינאריות מעומק 2 בלבד, ניתן לבצע סימולציה פולינומיאלית של מערכות ההוכחה רזולוציה, PC ו-PCR.

מחקרים בסיבוכיות של הוכחות אלגבריות ופסוקיות

תקציר

חקר הסיבוכיות של הוכחות הוא תחום המצוי בקו התפר בין תורת הסיבוכיות החישובית לבין הלוגיקה המתמטית והחישובית. במרכז התחום מצויה הבעיה של סיווג הטענות להן יש הוכחות קצרות או יעילות במערכות הוכחה פורמליות נתונות. בבואנו לעסוק בסיווג הוכחה כיעילה (feasible) עלינו לשקול בדרך כלל שני מרכיבים של ההוכחה: ראשית, את אורך או גודל ההוכחה, קרי, את מספר הסימנים המופיעים בסדרת ההוכחה; ושנית את "המושגים" בהם משתמשת ההוכחה - ובמובן הטכני, את מחלקת הסיבוכיות ממנה לקוחות שורות ההוכחה.
תיזה זו מהווה תרומה למחקר בסיבוכיות של הוכחות במובן הרחב, קרי, כתורה שעניינה בגדלים של הוכחות סימבוליות כלשהן. מרבית התיזה תעסוק בפיתוח ובחינה של הוכחות בעלות אופי אלגברי ומערכות המשלבות יחדיו הסקה אלגברית ובוליאנית. מעבר לכך, נקשור בין שאלות אודות מערכות ההוכחה שנציג לבין שאלות בסיבוכיות אלגברית.

סיבוכיות של הוכחות פסוקיות

המושג המרכזי של תורת הסיבוכיות של ההוכחות *הפסוקיות* הוא המושג של *מערכת הוכחה פורמלית* (או *אבסטרקטית*) כפי שהוגדר על-ידי Cook ו-Reckhow ([1979] Cook and Reckhow):

הגדרה: *מערכת הוכחה פורמלית היא אלגוריתם בעל זמן ריצה פולינומיאלי A המקבל כקלט נוסחה פסוקית F ומחרוזת π מעל אלפבית סופי (לכאורה את "ההוכחה של F"), כך שקיימת מחרוזת π המקיימת $A(F,\pi)=1$ אם ורק אם F היא טאוטולוגיה פסוקית.*

בהינתן הגדרה כללית זו למערכת הוכחה פסוקית, השאלה המרכזית בתחום היא כלהלן:

תהא A מערכת הוכחה פסוקית נתונה ותהא F טאוטולוגיה פסוקית נתונה; מהו הגודל הקטן ביותר של הוכחה ב-A של הטאוטולוגיה F ?

לשאלה זו יש חשיבות מכרעת הן בסיבוכיות של חישובים (כפי שהראו Cook ו- Reckhow, אם לכל מערכת הוכחה פורמלית קיימות טאוטולוגיות שאין להן הוכחות פולינומיאליות [בגודל הטאוטולוגיות], אזי **$\mathbf{NP \neq coNP}$**); הן בלוגיקה המתמטית ובחקר תורות חלשות של אריתמטיקה מסדר ראשון (ומעלה); והן בשאלות אודות יעילות של כלי הוכחה אוטומטיים. כמו כן, לשאלה זו קשר אמיץ, לכל הפחות מבחינת הטכניקות ורוח ההוכחות בתחום, עם שאלות בסיבוכיות של מעגלים בוליאניים.

תמצית

חקר הסיבוכיות של הוכחות הוא תחום מפותח שמרכזו בסיווג הטענות אותן ניתן להוכיח באמצעות הוכחות קצרות במערכת הוכחה פורמלית נתונה. המוטיבציות והשורשים של תחום זה באים הן מתורת הסיבוכיות החישובית והן מהלוגיקה המתמטית. תיזה זו עוסקת ברובה בפיתוח ובחינה של מערכות הוכחה בעלות אופי אלגברי ומערכות הוכחה המשלבות הסקה אלגברית ובוליאנית. בנוסף, התיזה מקשרת בין שאלות אודות מערכות ההוכחה שנציג לבין שאלות בסיבוכיות אלגברית. התוצאות המרכזיות בתיזה ניתנות לחלוקה לארבעה חלקים, כלהלן.

בחלק הראשון של התיזה אנו מציגים מערכות אלגבריות פסוקיות שעובדות עם נוסחאות מולטי-לינאריות. אנו מראים כי הוכחות מולטי-לינאריות המשתמשות בנוסחאות מולטי-לינאריות מעומק לכל היותר 3 מהוות מערכת הוכחה חזקה דיה כדי לאפשר סימולציה של (ואף הפרדה אקספוננציאלית מ-) מערכות הוכחה פסוקיות מקובלות אחרות. בנוסף לכך נראה קשר בין הוכחת חסמים תחתונים מפורשים על מעגלים אריתמטיים (כלליים) לבין הוכחת חסמים תחתונים על הוכחות מולטי-לינאריות.

בחלקה השני של התיזה אנו מציגים מערכות פסוקיות המשלבות היסקים אלגבריים ולוגיים כהרחבות של מערכת הרזולוציה. כל שורת-הוכחה במערכות אלו היא דיסיונקציה של משוואות לינאריות עם מקדמים שלמים הנכתבים בייצוג אונרי. נראה כי מערכות אלו - ואף תת-מערכות ממש שלהן - חזקות דיין כדי לאפשר הוכחות יעילות (קרי, בעלות גודל פולינומיאלי) של טענות קשות. מאידך, נוכיח חסמים תחתונים אקספוננציאליים על גודל ההוכחות של תת-מערכות אלו, עבור טענות מסוימות. מעבר לכך, נקשור את תת-המערכות הללו להוכחות מולטי-לינאריות מעומק 3.

חלק ממערכות ההוכחה האלגבריות אותן נציג הן מערכות *סמנטיות*, קרי, מערכות בהן כל שורת-הוכחה נובעת מקודמותיה במובן הסמנטי בלבד, אך לא בהכרח בצורה המפורשת בה נכתבת השורה (במילים אחרות, אין בהכרח אלגוריתם *דטרמיניסטי* יעיל המסוגל להכריע אם שורת ההוכחה אכן נובעת מקודמותיה). בחלקה השלישי של התיזה, נראה כיצד להפוך הוכחה אלגברית סמנטית להוכחה סינטקטית (קרי, כזו הניתנת לזיהוי בזמן פולינומיאלי). נעשה זאת באמצעות הצגה של מערכות הוכחה בהן כל הוכחה מהווה עדות לכך שנוסחה אריתמטית נתונה מחשבת את פולינום האפס. כללי ההיסק של מערכות אלו יהיו האקסיומות של חוג-הפולינומים. נוכיח חסם תחתון - הראשון מסוגו - על תת-מערכות של מערכת זו.

בחלק הרביעי והמסיים של התיזה, נציג מערכות הוכחה פסוקיות, כהרחבות של רזולוציה, שהן שלמות ונאותות עבור קבוצת הטאוטולוגיות הפסוקיות, *תחת ההבטחה שאם נוסחה אינה ספיקה הרי שיש לה "הרבה" השמות לא מספקות* (כאשר המדד "הרבה" מבוטא במפורש על-ידי פרמטר Λ כלשהו). נראה ששינוי גודל ההבטחה Λ מוביל להפרדה אקספוננציאלית (במקרה הממוצע) בין מערכות הוכחה בעלות הבטחות Λ גדולות יותר (קרי, שבר קבוע מסך כל ההשמות האפשריות: $\Lambda = \varepsilon \cdot 2^n$) לאלו עם הבטחות Λ קטנות יותר (קרי, $\Lambda = 2^{\delta n}$).

עבודה זו נעשתה בהדרכתם של

פרופ' **נחום דרשוביץ** (אוניברסיטת תל אביב) ופרופ' **רן רז** (מכון ויצמן למדע)

אוניברסיטת תל אביב
הפקולטה למדעים מדויקים ע"ש ריימונד ובברלי סאקלר
בית הספר למדעי המחשב

מחקרים בסיבוכיות של הוכחות אלגבריות ופסוקיות

חיבור לשם קבלת תואר "דוקטור לפילוסופיה"

מאת

עידו צמרת

בהנחיית

פרופ' נחום דרשוביץ ופרופ' רן רז

הוגש לסנאט של אוניברסיטת תל-אביב
אב תשס"ח, אוגוסט 2008