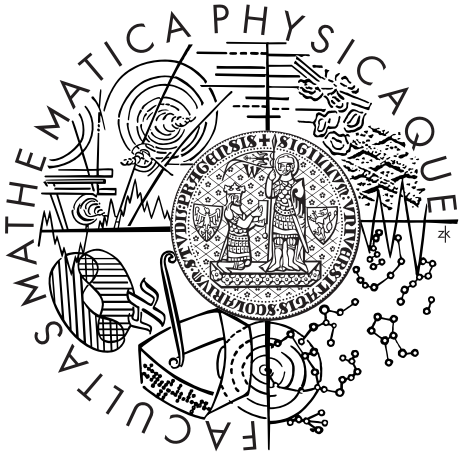
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Sebastian Müller

On the Power of Weak Extensions of V^0

Katedra Algebry

Supervisor of the doctoral thesis: Prof. RNDr. Jan Krajíček, DrSc.
Study programme: Matematika
Specialization: Matematická Logika

Prague 2012

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: O síle slabých rozšírení teorie $\mathbf{V}^0$

Autor: Sebastian Müller

Katedra: Katedra Algebry

Vedoucí disertační práce: Prof. RNDr. Jan Krajíček, DrSc., Katedra Algebry.

Abstrakt: V predložené disertacní práci zkoumáme sílu slabých fragmentu aritmetiky. Činíme tak jak z modelově-teoretického pohledu, tak z pohledu důkazové složitosti. Pohled skrze teorii modelu naznačuje, že malý iniciální segment libovolného modelu omezené aritmetiky bude modelem silnější teorie. Jako příklad ukážeme, že každý polylogaritmický řez modelu $\mathbf{V}^0$ je modelem **VNC**. Užitím známé souvislosti mezi fragmenty omezené aritmetiky a dokazatelností v různých důkazových systémech dokážeme separaci mezi rezolucí a TC^0-Frege systémem na náhodných 3CNF-formulích s jistým poměrem počtu klauzulí vůci počtu proměnných. Zkombinováním obou výsledků dostaneme slabší separační výsledek pro rezoluci a Fregeho důkazové systémy omezené hloubky.

Klíčová slova: omezená aritmetika, důkazová složitost, Fregeho důkazový systém, Fregeho důkazový systém omezené hloubky, rezoluce

Title: On the Power of Weak Extensions of $\mathbf{V}^0$

Author: Sebastian Müller

Department: Department of Algebra

Supervisor: Prof. RNDr. Jan Krajíček, DrSc., Department of Algebra.

Abstract: In this thesis we investigate the power of weak fragments of arithmetic. We do this from a model theoretic and also from a proof complexity perspective. From a model theoretic point of view it seems reasonable that a small initial segment of any model of bounded arithmetic is a model of a stronger theory. We exemplify this by showing that any polylogarithmic cut of a model of $\mathbf{V}^0$ is actually a model of **VNC**. Exploiting a well-known connection between fragments of bounded arithmetic and provability in various proof systems, we show a separation result between Resolution and the TC^0-Frege proof system on random 3CNF within a certain clause-to-variable ratio. Combining both results we can also conclude a weaker separation result for Resolution and bounded depth Frege systems.

Keywords: Bounded Arithmetic, Proof Complexity, Frege proof system, bounded depth Frege proof system, Resolution.

Acknowledgements

I want to thank my supervisor Jan Krajíček, who was not only a great source of inspiration and support, but also never failed to create a funny and relaxed atmosphere, which made working together a great experience. I also want to thank Pavel Pudlák, Emil Jeřábek and Neil Thapen for many helpful discussions, hints and the one or the other enlightening words. I am happy to have met numerous friends and colleagues in Prague, among which are Iddo Tzameret, Ján Pich, Michal Garlík and Zi Chao Wang, who sometimes shared a thought and more often a beer. I also want to thank Olaf Beyersdorff and Johannes Köbler for introducing me to the field of proof complexity. My research was supported by the Marie Curie Initial Training Network -MALOA-, PITN-GA-2009-238381.

Contents

Preface

In this thesis we will investigate very weak fragments of arithmetic and their connection to provability in various propositional proof systems. Bounded Arithmetic has been introduced by Parikh in [71] and was widely studied as a feasible subtheory of Peano Arithmetic ever since. Parikh introduced the system $\mathbf{I\Delta}_0$, which originally consisted of Robinson's Arithmetic Q, together with induction over Δ_0 formulas. Subsequently, extensions of $\mathbf{I\Delta}_0$ were considered that postulated the existence of a function needed for the coding of sequences. In [20], Sam Buss developed a hierarchy of theories S^i_2 that subdivides that extension of $\mathbf{I\Delta}_0$ and pinpointed the strength of these theories by proving a strong relation between definability with respect to them and computability of the appropriate properties in levels of the Polynomial Hierarchy. On the other hand provability of certain properties also corresponds to efficient provability in propositional proof systems. The latter correspondence is the one we will be most interested in. The first chapter summarizes known background results and can be skipped by a reader who is familiar with the subject. A broader background is included for completeness and for a quick reference as Appendix C. The original research is presented in Chapter 2 through 5. The thesis can be summarized as follows.

In Chapter 1 we introduce the background necessary for our results. That is, we start with a brief overview of Proof Complexity and Bounded Arithmetic, for which the textbooks by Krajíček [57] and Cook and Nguyen [29] are excellent sources. Then, in Section 1.3 we give an overview of the theory $\mathbf{V}^0$ and some of its extensions, namely $\mathbf{VTC}^0$ and $\mathbf{VNC}^k$. We will recapture some properties of $\mathbf{VTC}^0$, especially its connections to the propositional proof system TC^0-Frege and some functions available in that theory. We will then continue to introduce $\mathbf{VNC}^k$ and show that each such theory extends $\mathbf{VTC}^0$, thus making these functions also available there.

In Chapter 2 we give an overview of the proof of the main result of M. and Tzameret [67], which is given in full in Appendix A. That is, we will show that in TC^0-Frege we can exploit a result by Feige, Kim and Ofek [37] to efficiently prove the unsatisfiability of certain random 3CNF, which is known to be impossible in Resolution. We therefore obtain a separation result on randomly generated formulas, showing a fundamental difference in the strength of these two proof systems.

In Chapter 3 we return to the theory $\mathbf{V}^0$, but this time perceive it from a model theoretic perspective. We will define the notion of an initial cut and then restate the main result of [66] that a certain kind of Turing computability of a property leads to its definability in a small initial cut. We will sketch the argument, which is fully given as Appendix B. This implies that any such an initial cut is a model of a stronger theory and we will finally strengthen this observation by exploiting a different algorithm than the one that was initially used.

In Chapter 4 we will further explore the computational strength of $\mathbf{VTC}^0$. This chapter contains also the original motivation for the results of Chapter 3, which is an idea of how to prove the non-automatizability of Resolution, together with an explanation, why this approach might be bound to fail to obtain the desired result. The whole chapter is rather a report on work in progress. Yet

we have some partial results that may be of some interest, so I opted to present them.

In Chapter 5 we will discuss the presented results and state some directions for further investigation.

Appendix A contains the article [67] in the form it was accepted at LICS 2012. In it, Iddo Tzameret and I show that reasoning in $\mathbf{VTC}^0$ allows for the use of some results from linear algebra by approximating the real numbers by close enough rationals with standardized denominators. This lets us exploit the result from [37] to conclude that certain random 3CNF are efficiently refutable in TC^0-Frege, while their Resolution refutations are exponentially long.

Appendix B is the submitted article [66], showing that a version of Nepomnjascij's Theorem holds in small initial cuts of models of $\mathbf{V}^0$. That is, if we can show that a given property can be computed by a Turing machine in $\mathsf{TimeSpace}(n^c, n^\epsilon)$ for some $c \in \mathbb{N}$ and $\epsilon < 1$, we immediately get the Δ_0^B-definability of every set with that property in the initial cut. This implies a subexponential simulation result of Frege systems by bounded depth Frege systems, which was first proved by Filmus, Pitassi and Santhanam in [39].

In Appendix C we give a brief overview of propositional and first-order logic and their elementary proof theory, as well as to complexity theory, specifically also circuit complexity and bounded arithmetic and proof complexity. The exposition is supposed to be self-contained, but due to its brevity, the topics are more accessible from textbooks, such as Barwise [8], Buss [22], Ebbinghaus, Flum and Thomas [35], and Pudlák [80] for Sections C.1.1 and C.1.2. For Section C.2 we refer the reader to Arora and Barak [5]. That book also contains most of the information for Section C.2.1, a more thorough treatment is in Vollmer [90], though.

1. Preliminaries

We assume familiarity with the basic notions of logic and complexity theory and will only recapture a few facts which are of importance to us. A longer, though still brief, survey of logics and complexity theory is available for the convenience of a reader not familiar with logics as Appendix C.

1.1 Proof Complexity

In their seminal paper "The Relative Efficiency of Propositional Proof Systems" [30] Cook and Reckhow examined various propositional calculi, such as Resolution, truth table, Frege or Gentzen's PK, in a general form. They concentrated on the property that there is an efficient way of validating a proof with respect to any such system and embodied this in the following definition.

Definition 1.1. A *propositional proof system* is any polynomial time computable function P that maps strings onto TAUT. For any tautology φ a *P-proof* of φ is any string $\pi \in \Sigma^*$ that gets mapped via P to φ.

As we will see soon the length of proofs in various propositional proof systems is a very interesting subject of investigation. For brevity we will often write *proof system* instead of propositional proof system if what we are talking about is clear from the context. A propositional proof system P is *polynomially bounded* iff for every $\varphi \in \mathsf{TAUT}$ there exists a P-proof $\pi \in \Sigma^*$ with $|\pi| \leq p(|\varphi|)$ for some fixed polynomial p. A proof system P *simulates* a proof system Q (in symbols $P \geq Q$) iff there exists a polynomial p such that for every $\pi_Q \in \Sigma^*$ there exists a $\pi_P \in \Sigma^*$ with $|\pi_P| \leq p(|\pi_Q|)$ and $P(\pi_P) = Q(\pi_Q)$. A proof system is *optimal* iff it simulates every other proof system. Two proof systems P and Q are *equivalent* ($P \equiv Q$) iff they mutually simulate each other.

Propositional proof systems are an interesting concept, also from the perspective of Complexity Theory. This is evident from the following observation from [30].

Theorem 1.2 ([30])**.** $\mathsf{NP} = \mathsf{coNP}$ *iff there exists a polynomially bounded propositional proof system.*

Proof. For the "if" direction observe that TAUT is coNP-complete. It therefore suffices to give an NP procedure for TAUT. Let P be a polynomially bounded proof system with bounding polynomial p, then we can construct a nondeterministic Turing machine A deciding TAUT as follows. On input φ, A nondeterministically guesses a P-proof π of length at most $p(|\varphi|)$ and then simulates the Turing machine computing P to verify that $P(\pi) = \varphi$. This is clearly polynomial in the input length.

For the "only if" direction, assume that A is a nondeterministic polynomial time Turing machine deciding TAUT. For every input φ we let w_φ be the nondeterministic witness that A guesses and then verifies. We can require that w_φ also contains φ at an explicit position. We define a propositional proof system P by the following Turing machine M. On input w_φ, M simulates A deterministically on input φ and outputs φ if A would accept. Otherwise it outputs a predefined

tautology, such as $p \vee \neg p$. The resulting function obviously has $TAUT$ as its range and is onto. □

Another interesting point of research is the question: "How hard is it to find a proof in a given proof system P?" We call a proof system *automatizable* iff there is an algorithm that, given a tautology φ returns a P-proof of φ using at most a polynomially, in the length of the shortest P-proof for φ, many steps. The question of the automatizability of proof systems is obviously an interesting subject, unfortunately it turns out that most stronger ones are not (under mild assumptions). It is still interesting, though, how the picture would look like when we weaken the time constraint (e.g. by allowing quasipolynomial time computations).

1.1.1 Some Important Proof Systems and Their Interrelation

We will focus on a few important propositional proof systems. The choice of these systems is due to the focus of this thesis and is by no means exhaustive. The systems we will have a closer look at are Resolution, bounded depth Frege, TC^0-Frege and Frege. We will elaborate on their computational aspects here, most of the definitions can be found in Appendix C.

Resolution

Resolution (see Section C.1.1 in Appendix C) is a very simple proof system, as it only has one rule to apply. All that remains for choice is to pick the clauses and variables to resolve on. This property makes Resolution very interesting for proof search algorithms, because the number of possibilities seems to be much more feasible than in other proof systems. Thus, various algorithms have been devised. The oldest certainly is DPLL, named after its inventors Davis, Putnam, Logemann and Loveland (see [31] and [32]), which is an algorithm that corresponds to a treelike Resolution refutation and is still a vital part of most of today's satisfiability solvers.

Albeit its striking success in application, Resolution suffers from several serious drawbacks concerning its efficiency, i.e. its proof complexity, which we will explore in this section. Time will tell, whether these drawbacks are outweighed by its simplicity or not. Up to now, research into proof search algorithms using stronger proof systems has been limited, though it is becoming stronger and we will be able to make a clearer justification for or against the use of Resolution based proof search algorithms in the future.

What we do know today is that Resolution is not only rather inefficient concerning proofs of combinatorial principles, but even concerning random statements. An example for a combinatorial principle is the Weak Pigeonhole Principle, for random statements it suffices to consider random kCNF with a certain clause to variable ratio.

For $m > n$, the weak Pigeonhole Principle PHP_n^m is the statement that there is no injective mapping from a set with m elements (the "pigeons") to one with n (the "holes"). The Pigeonhole Principle is the hardest case of the weak Pigeonhole

Principle and is given as PHP_n^{n+1}. The negations of these principles are clearly unsatisfiable and can be given as CNFs in the following way.

For the set $\{p_{ij} : i \in m, j \in n\}$ we let for all i, i_1, i_2, j

$$Q_i := \bigvee_{j \in n} p_{ij}, and$$

$$Q_{i_1,i_2,j} := (\bar{p}_{i_1 j} \vee \bar{p}_{i_2 j}).$$

The negation $\neg\mathsf{PHP}_n^m$ of the Pigeonhole Principle is

$$\bigwedge_{i \in m} Q_i \wedge \bigwedge_{i_1 \neq i_2 \in m, j \in n} Q_{i_1,i_2,j}.$$

The intended meaning is that p_{ij} holds iff pigeon i gets mapped to hole j and therefore Q_i states that pigeon i gets a hole, while $Q_{i_1,i_2,j}$ states that pigeon i_1 and pigeon i_2 will not both be on hole j. This principle is a standard example showing the different strengths of various proof systems and we encounter it several times in this chapter.

Theorem 1.3 ([50, 24, 81, 82])**.** *Resolution does not have subexponential proofs of the Weak Pigeonhole Principle* PHP_n^m*.*

The result for the Pigeonhole Principle was established by Haken in [50]. For the weak version lower bounds were first proved by Buss and Turan [24], whose results were subsequently improved by Raz [81] and by Razborov [82].

On random kCNF, Resolution does not perform much better. We call a kCNF φ random with m clauses and n variables, iff φ is produced by randomly (uniformly distributed) choosing m clauses (with repetitions) from the $2^k \cdot \binom{n}{k}$ available ones. Chvátal and Szemerédi [25] showed that such random kCNF are mostly unsatisfiable, when the number of clauses exceeds the number of variables by some constant factor:

Theorem 1.4 ([25])**.** *Let φ be a random kCNF with m clauses and n variables. If $\frac{m}{n} > 2^k \cdot \ln(2)$, then, with high probability, φ is unsatisfiable.*

We know, however, that Resolution cannot witness this unsatisfiability efficiently unless the clause-to-variable ratio gets much worse. The following is a slight improvement of Ben-Sasson and Wigderson [15] over a result by Chvátal and Szemerédi [25]. It is stated for 3CNF here, but can easily be extended to kCNF for arbitrary $k > 2$.

Theorem 1.5 ([25, 15])**.** *Let φ be a random propositional 3CNF with m clauses and n variables. Then, with probability approaching 1, Resolution has no subexponential refutation of φ if the clause to variable ration $\frac{m}{n} < n^{1,5-\epsilon}$.*

The question of the automatizability has been settled negatively for Resolution by Alekhnovich and Razborov [3] under a complexity theoretic assumption.

Theorem 1.6 ([3])**.** *If $\mathsf{W}[\mathsf{P}] \not\subset co - \mathsf{FPR}$, then neither Resolution nor tree-like Resolution is automatizable.*

It is still possible, though, that Resolution allows for quasipolynomial automatizability.

Bounded Depth Frege

Bounded depth Frege or AC^0-Frege is a proof system that is defined similar to the Frege system, with the exception that the cut rule can only be applied to formulas with a constant depth. These systems are still rather weak, but can simulate Resolution (see [57]).

Theorem 1.7. *B.d. Frege simulates Resolution.*

On the other hand, there are versions of the weak Pigeonhole Principle that are efficiently provable in b.d. Frege. The following was proved by Paris, Wilkie and Woods [76].

Theorem 1.8 ([76])**.** *B.d. Frege has subexponential proofs of* PHP_n^m *for* $m = 2n$.

As we shall see later in Chapter 3, bounded depth Frege systems also outperform Resolution on random 3CNF with a certain clause-to-variable ratio. Though, as far as we know, only with subexponential speedup. Unfortunately, these systems are still to weak to prove the full Pigeonhole Principle.

Theorem 1.9 ([1, 62, 78, 9])**.** *B.d. Frege systems do not admit subexponential proofs of* PHP_n^{n+1}.

The first result of this kind was established by Ajtai [1], who used a relation between propositional proof systems and bounded arithmetic in conjunction with a model-theoretic analysis of the problem. He was able to show that no polynomial size proofs can exist. We will explore the relation between bounded Arithmetic and proof systems in the Section 1.2. Subsequently, Ajtai's lower bound was strengthened to the above theorem in [62, 78, 9].

B.d. Frege systems already outperform Resolution drastically on the weak Pigeonhole Principle, since Paris, Wilkie and Woods [76] showed that there are quasipolynomial proofs of PHP_n^m if $m > n + \frac{n}{(\log n)^k}$ for some k. In contrast to this, Buresh-Oppenheim, Beame, Pitassi, Raz and Sabharwal [19] showed that this bound cannot be strengthened to polynomial proofs if m is at the lower end of the interval, i.e. if $m = n + \frac{n}{(\log n)^k}$.

Bounded depth Frege systems constitute a strict hierarchy, depending on the depth of formulas they allow the cut-rule for. A system having the cut-rule for formulas of depth at most d cannot simulate, but is simulated by, a system allowing the cut-rule for formulas of depth $d + 1$. See e.g. [57] for a proof.

Under moderate cryptographic assumptions, there is also no automatization algorithm for bounded depth Frege. That result is an adaption of an argument from Krajíček and Pudlák [61] to an NP pair for the Diffie-Hellman key exchange protocol by Bonet, Domingo, Gavaldà, Maciel and Pitassi [17].

Theorem 1.10 ([17])**.** *If the Diffie-Hellman key exchange protocol is secure then bounded depth Frege is not automatizable.*

TC^0-Frege

We will first define the notion of TC^0 formulas, a generalization of formulas by a threshold primitive, and then define the propositional proof system TC^0-Frege

as a sequent calculus operating on such formulas. We will follow the exposition from [29]. The system we give is only one of many possibilities to define such proof systems (see e.g. [18] for a polynomially-equivalent definition).

The class of TC^0 formulas consists basically of unbounded fan-in constant depth formulas with $\wedge, \vee, \neg$ and threshold gates. Formally, we define:

Definition 1.11 (TC^0 formula)**.** A TC^0 formula is built from

(i) propositional constants $\bot$ and $\top$,

(ii) propositional variables p_i for $i \in \mathbb{N}$,

(iii) connectives $\neg$ and Th_i, for $i \in \mathbb{N}$.

Items (i) and (ii) constitute the *atomic formulas*. TC^0 formulas are defined inductively from atomic formulas via the connectives:

(a) if A is a formula, then so is $\neg A$ and

(b) for $n > 1$ and $i \in \mathbb{N}$, if $A_1, \ldots, A_n$ are formulas, then so is $\mathsf{Th}_i A_1 \ldots A_n$.

The *depth* of a formula is the maximal nesting of connectives in it and the *size* of the formula is the total number of connectives in it.

For the sake of readability we will also use parentheses in our formulas, though they are not necessary. The semantics of the *Threshold Connectives* Th_i are as follows. $\mathsf{Th}_i(A_1, \ldots, A_n)$ is true if and only if at least i of the A_k are true. Therefore we will abbreviate $\mathsf{Th}_i(A_1, \ldots, A_i)$ as $\bigwedge_{k \leq i} A_k$ and $\mathsf{Th}_1(A_1, \ldots, A_i)$ as $\bigvee_{k \leq i} A_k$. Moreover we let $\mathsf{Th}_0(A_1, \ldots, A_n) = \top$ and $\mathsf{Th}_i(A_1, \ldots, A_n) = \bot$, for $i > n$.

The following is the sequent calculus TC^0-Frege.

Definition 1.12 (TC^0-Frege)**.** A TC^0-Frege proof system is a sequent calculus with the axioms

$$A \longrightarrow A, \qquad \bot \longrightarrow, \qquad \longrightarrow \top,$$

where A is any TC^0 formula, and the following derivation rules:

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ \text{(Weaken Left)} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}\ \text{(Weaken Right)}$$

$$\frac{\Gamma_1, A_1, A_2, \Gamma_2 \longrightarrow \Delta}{\Gamma_1, A_2, A_1, \Gamma_2 \longrightarrow \Delta}\ \text{(Exchange Left)} \qquad \frac{\Gamma \longrightarrow \Delta_1, A_1, A_2, \Delta_2}{\Gamma \longrightarrow \Delta_1, A_2, A_1, \Delta_2}\ \text{(Exchange Right)}$$

$$\frac{\Gamma, A, A \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ \text{(Contract Left)} \qquad \frac{\Gamma \longrightarrow A, A, \Delta}{\Gamma \longrightarrow A, \Delta}\ \text{(Contract Right)}$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma, \neg A \longrightarrow \Delta}\ (\neg \text{ Left}) \qquad \frac{\Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \neg A, \Delta}\ (\neg \text{ Right})$$

$$\frac{A_1, \ldots, A_n, \Gamma \longrightarrow \Delta}{\mathsf{Th}_n A_1 \ldots A_n, \Gamma \longrightarrow \Delta}\ \text{(All Left)} \qquad \frac{\Gamma \longrightarrow A_1, \Delta, \ldots, \Gamma \longrightarrow A_n, \Delta}{\Gamma \longrightarrow \mathsf{Th}_n A_1 \ldots A_n, \Delta}\ \text{(All Right)}$$

$$\frac{A_1, \Gamma \longrightarrow \Delta, \ldots, A_1, \Gamma \longrightarrow \Delta}{\mathsf{Th}_1 A_1 \ldots A_n, \Gamma \longrightarrow \Delta}\ \text{(One Left)} \qquad \frac{\Gamma \longrightarrow A_1, \ldots, A_n, \Delta}{\Gamma \longrightarrow \mathsf{Th}_1 A_1 \ldots A_n, \Delta}\ \text{(One Right)}$$

$$\frac{\mathsf{Th}_i A_2 \ldots A_n, \Gamma \longrightarrow \Delta \qquad \mathsf{Th}_{i-1} A_2 \ldots A_n, A_1, \Gamma \longrightarrow \Delta}{\mathsf{Th}_i A_1 \ldots A_n, \Gamma \longrightarrow \Delta}\ (\mathsf{Th}_i \text{ Left})$$

$$\frac{\Gamma \longrightarrow \mathsf{Th}_i A_2 \ldots A_n, A_1, \Delta \qquad \Gamma \longrightarrow \mathsf{Th}_{i-1} A_2 \ldots A_n, \Delta}{\Gamma \longrightarrow \mathsf{Th}_i A_1 \ldots A_n, \Delta}\ (\mathsf{Th}_i \text{ Right})$$

$$\frac{\Gamma \longrightarrow A, \Delta \qquad \Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}\ \text{(Cut)}$$

for arbitrary TC^0 formulas A_i and sets Γ, Δ of TC^0 formulas. The intended meaning of $\Gamma \longrightarrow \Delta$ is that the conjunction of the formulas in Γ implies the disjunction of the formulas in Δ. A TC^0-Frege proof of a formula φ is a sequence of sequents $\pi = (S_1, \ldots, S_k)$ such that $S_k = \longrightarrow \varphi$ and every sequent in it is either an axiom or was derived from previous lines by a derivation rule. The *size* of the proof π is the total size of all formulas in its sequents. The *depth* of the proof π is the maximal depth of a formula in its sequents. A TC^0-Frege proof of *a family of formulas* $\{\varphi_i : i \in \mathbb{N}\}$ is a family of sequences $\{(S_1^i, \ldots, S_{k^i}^i) : i \in \mathbb{N}\}$, where each S_j^i is a TC^0 formula that can be derived from some S_k^i for $k < j$ using the above rules, such that $S_{k^i}^i = \longrightarrow \varphi_i$, and there is a *common constant* c *bounding the depth of every formula in all the sequences.*

Proposition 1.13. The proof system TC^0-Frege is sound and complete for formulas of any fixed depth d. That is, every formula A proven in the above way is a tautology and every tautology of depth d can be derived by proofs in the above sense.

We will now explore the proof theoretic strength of TC^0-Frege. As we will see in Chapter A, it outperforms Resolution drastically on random 3CNF of a certain clause-to-variable ratio. Also, it is the weakest proof system we consider that proves the Pigeonhole Principle PHP_n^{n+1}. See [29] for a proof that exploits the relation between arithmetic and proof systems.

Theorem 1.14. TC^0*-Frege has polynomial size proofs of the Pigeonhole Principle.*

Obviously it simulates bounded depth Frege systems.

TC^0-Frege systems are not automatizable, if factoring for Blum integers is hard. This result is due to Bonet, Pitassi and Raz [18] and builds on the interpolation argument first presented by Krajíček and Pudlák [61].

Theorem 1.15 ([18])**.** *If factoring of Blum integers is hard, then* TC^0*-Frege is not automatizable.*

Frege

We will now elaborate on the proof theoretic strength of Frege systems (if necessary, see Section C.1.1 for a defintion). The first thing that is worth mentioning is that, from the perspective of Proof Complexity, there is no need to distinguish between various Frege systems because all Frege systems as we have defined them, mutually simulate each other, as was shown by Reckhow [83].

Theorem 1.16 ([83])**.** *Let* F_1 *and* F_2 *be Frege systems, then* $F_1 \equiv F_2$*.*

Additonally, as can be seen by observing the provability of the Reflection Principles for TC^0-Frege, i.e. the statement that the proofs are correct, in Frege systems, we obtain that Frege systems simulate TC^0-Frege.

Theorem 1.17. *Frege simulates* TC^0*-Frege.*

As far as we know, Frege systems constitute an extremely strong family of proof systems. No superpolynomial lower bounds are known to date. The presumed strength of Frege systems is backed by the fact that they efficiently prove the Pigeonhole Principle, as was shown by Buss [22] and also follows from the above simulation of TC^0-Frege.

Theorem 1.18 ([22])**.** *There are polynomial size Frege proofs of the Pigeonhole Principle* PHP_n^{n+1}*.*

Not surprisingly, this strong family of proof systems is not automatizable under the same mild cryptographic assumptions, as for TC^0-Frege. That is, Bonet, Pitassi and Raz [18] showed the following.

Theorem 1.19 ([18])**.** *Unless factoring of Blum integers is feasible, Frege systems are not automatizable.*

Gentzen's PK

From a proof complexity perspective there is not much sense in considering PK separately from Frege, as both systems can be shown to be equivalent (see e.g. [57]). We will therefore often use PK for its nice properties when we want to argue about arbitrary Frege systems.

Theorem 1.20. *Gentzen's* PK *is polynomially equivalent to Frege.*

We will conclude this section by recapturing the relations of the proof theoretic strengths of the proof systems of interest for us:

$$Res \lneq b.d.Frege \lneq \mathsf{TC}^0 - Frege \leq Frege \equiv PK.$$

1.2 Bounded Arithmetic

In Arithmetic one wants to capture the "real" world of natural numbers by logical means, given by a calculus, such as LK or FC, and a set of axioms. The first such characterizations are due to Grassmann [48], Frege [40], Dedekind [33] and Peano [77]. Peano's way of characterizing numbers prevailed until today and we will make his characterization our starting point. He considered the constant 0 and the successor function S as logical primitives and first stated the axiom of full induction

$$(FI) \text{ For every predicate } P: \ P(0) \wedge \forall x(P(x) \to P(S(x))) \to \forall x P(x).$$

This axiomatization is strong enough to define the natural numbers up to isomorphism.

Theorem 1.21. *Every model of arithmetic with full induction is isomorphic to the natural numbers.*

Proof. Assume that it would not be the case and let M be a non-isomorphic model. Then, since M is a model of induction there must be a subset of M isomorphic to $\mathbb{N}$. Taking this subset as the predicate P in the definition of induction leads to a contradiction. □

As we have seen, arithmetic with set induction is an adequate way of representing the natural numbers. However, from a logical perspective it is rather strong, as we claim statements about arbitrary subsets. A logically more accessible approach is to limit the induction to definable sets, which leads us to first-order induction

$$(IND) \text{ For every formula } \varphi: \ \varphi(0) \wedge \forall x(\varphi(x) \to \varphi(S(x))) \to \forall x \varphi(x).$$

This is a more feasible way of representing the natural numbers, but it comes at a cost.

Theorem 1.22. *There are models of arithmetic with first-order induction that are not isomorphic to the natural numbers.*

Proof. We let

$$\underline{n} = \underbrace{S(\dots S(0)\dots)}_{n \text{ times}}$$

be the nth numeral. Let c be any constant symbol. Now, consider arithmetic together with $\{c > \underline{n}) : n \in \mathbb{N}\}$. By the Compactness Theorem there exists a model M of that theory, but it obviously cannot be $\mathbb{N}$. □

First-order Arithmetic is a theory with strong proof theoretic implications and thus well deserves to be explored on its own merit. We, however, will concern ourselves with weaker versions of arithmetic, since we can relate them to computations, as we shall soon see. This leads us to the definition of Bounded Arithmetic, where we restrict the induction axiom to bounded formulas. We call a quantifier *bounded* iff it is of the form $\exists x(x < t \wedge \varphi)$ or $\forall x(x < t \to \varphi)$ for some term t not containing x. We call a formula bounded iff all its quantifiers

are and we call a theory bounded, iff all the formulas it contains are. As a lot of elementary things that were definable before, are not definable with respect to bounded induction on the base of Robinson Arithmetic, we will now extend the language to contain symbols for addition, multiplication, ordering and the constant 1. The first such theory that springs to mind is presumably $\mathbf{I\Delta}_0$, which allows full induction for bounded formulas and is axiomatized as follows.

Basic 1. $x + 1 \neq 0$	**Basic 2**. $x + 1 = y + 1 \rightarrow x = y$
Basic 3. $x + 0 = x$	**Basic 4**. $x + (y + 1) = (x + y) + 1$
Basic 5. $x \cdot 0 = 0$	**Basic 6**. $x \cdot (y + 1) = (x \cdot y) + x$
Basic 7. $(x \leq y \wedge y \leq x) \rightarrow x = y$	**Basic 8**. $x \leq x + y$

Δ_0**-Ind**. $\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(x+1)) \rightarrow \forall x \varphi(x)$
for any bounded formula φ.

The models of $\mathbf{I\Delta}_0$ have various interesting properties (see for example [29]).

Proposition 1.23. Every model of $\mathbf{I\Delta}_0$ is the non-negtive part of a commutative, discretely-ordered semi-ring.

Parikh [71] proved the following interesting result.

Theorem 1.24 (Parikh)**.** *Let φ be a Δ_0^b-formula. If*

$$\mathbf{I\Delta}_0 \vdash \forall x \exists y \varphi(x, y),$$

then there exists a term t not containing y such that

$$\mathbf{I\Delta}_0 \vdash \forall x \exists y < t \varphi(x, y).$$

On the other hand, in models of $\mathbf{I\Delta}_0$ we are not able to tell the proper sizes of sets, as Ajtai showed in [1]. More precisely, let $\mathbf{I\Delta}_0(f)$ denote $\mathbf{I\Delta}_0$ with an additional function symbol f and induction extended to also allow use of f. Then $\mathbf{I\Delta}_0(f)$ does not disprove that f maps a set with $n+1$ elements injectively into a set with n elements.

Proposition 1.25. $\mathbf{I\Delta}_0(f) \nvdash PHP_n^{n+1}$.

Another problem with $\mathbf{I\Delta}_0$ is that it does not allow for coding of arbitrary sequences, as for example the code of the sequence of all numbers smaller than x is exponential in x. Another problem arises with substitution in strings. This is a consequence of Parikh's Theorem as the bounds given by the terms are polynomial and therefore are too small to allow for the aforementioned encodings. A non-trivial result by Bennett [11] shows that, although exponentiation is not definable in $\mathbf{I\Delta}_0$, its graph is. Moreover, by a result by Paris and Dimitracopoulos [72], $\mathbf{I\Delta}_0$ can prove several of its properties, especially concerning its recursive definition:

$$x^0 = 1 \text{ and } x^{y+1} = x \cdot x^y.$$

Pudlák [79] gave a new proof of this.

This at least ensures that $\mathbf{I\Delta}_0$ can talk about functions like $|\cdot|$ and codes of finite sequences. To allow for a more natural way of coding sequences we have to strengthen the theory, though. What is needed are numbers of size $2^{|x|^k}$ for arbitrary standard k. To do so, one can add an axiom, ensuring the existence of such numbers. Let the axiom Ω_1 be defined by $\forall x \exists y.\ y = x^{|x|}$ (this has to be perceived as talking about the graph of exponentiation). Then $\mathbf{I\Delta}_0 + \Omega_1$ is sufficient to speak about sequences and do some standard operations on strings.

A natural subdivision of $\mathbf{I\Delta}_0 + \Omega_1$ in a richer language are Buss's theories S_i^j and T_i^j, which nicely correspond to computability in certain complexity classes. We will, however, not go into detail, as the classes we are interested in are very weak and in this case, a two-sorted approach is more convenient. We will note, though, that the two-sorted approach below and the classical by Buss are equivalent as there is a canonical isomorphism between models of one theory and models of the other; the so-called $RSUV$-isomorphism (see [29] Chapter VIII or [57] Chapter 5).

To be able to make up a sensible weaker theory it is necessary to strengthen the language, as not everything can be proven from properties of $0, 1$ and $+, \cdot, \leq$ in the absence of strong induction. The primitives we will enhance our language with will be $0, 1, +, \cdot, |\cdot|, =_1, =_2, \leq, \epsilon$ and the "real world" we will interpret these primitives in will be $\mathbb{N}_2$, the two sorted model consisting of the natural numbers as the first sort and all finite subsets of natural numbers as its second sort universe. The constants $0, 1$ will have the natural interpretations in the first-sort universe, as do the functions $+$ and $\cdot$. The function $|\cdot|$ ranges from finite sets to numbers and, given a finite set, returns its largest element plus 1. This roughly represents the length of a sequence representing the characteristic function of the set. The relations $=_1$ and $=_2$ represent equality between numbers and sets, respectively and the relation $\leq$ is the usual ordering of the numbers. Finally, the relation ϵ is the elementhood relation between numbers and finite sets of numbers. We will call the above language of two-sorted arithmetic $\mathcal{L}_A^2 := \{0, 1, +, \cdot, |\cdot|, =_1, =_2, \leq, \epsilon\}$.

The weakest axiomatization for two-sorted arithmetic we will consider is $\mathbf{V}^0$, which we will define in the following chapter.

1.3 The Theory $\mathbf{V}^0$ and its Extensions

In this Chapter we will define the theory $\mathbf{V}^0$ and some of its extensions. We will also prove some basic properties and connections to Complexity Theory and Proof Complexity. We will follow the outline in [29].

As we have briefly sketched in Section 1.2, we are working in

$$\mathcal{L}_A^2 = \{0, 1, +, \cdot, |\cdot|, =_1, =_2, \leq, \epsilon\}.$$

We denote the first-sort (number) variables by lower-case letters $x, y, z, ...$, and the second-sort (string) variables by capital letters $X, Y, Z, ...$ We build formulas in the usual way, using two sorts of quantifiers: number quantifiers and string quantifiers. A number quantifier is said to be *bounded* if it is of the form $\exists x(x \leq t \wedge \dots)$ or $\forall x(x \leq t \rightarrow \dots)$, respectively, for some number term t that does not contain x. We abbreviate $\exists x(x \leq t \wedge \dots)$ and $\forall x(x \leq t \rightarrow \dots)$ by $\exists x \leq t$ and

$\forall x \leq t$, respectively. A string quantifier is said to be *bounded* if it is of the form $\exists X(|X| \leq t \wedge \dots)$ or $\forall X(|X| \leq t \rightarrow \dots)$ for some number term t that does not contain X. We abbreviate $\exists X(|X| \leq t \wedge \dots)$ and $\forall X(|X| \leq t \rightarrow \dots)$ by $\exists X \leq t$ and $\forall X \leq t$, respectively. A formula is in Σ_0^B or Π_0^B if it uses no string quantifiers and all number quantifiers are bounded. A formula is in Σ_{i+1}^B or Π_{i+1}^B if it is of the form $\exists X_1 \leq t_1 \dots \exists X_m \leq t_m \psi$ or $\forall X_1 \leq t_1 \dots \forall X_m \leq t_m \psi$, where $\psi \in \Pi_i^B$ and $\psi \in \Sigma_i^B$, respectively, and t_i does not contain X_i, for all $i = 1, \dots, m$. We write $\forall\Sigma_0^B$ to denote the universal closure of Σ_0^B. (i.e., the class of Σ_0^B-formulas that possibly have (not necessarily bounded) universal quantifiers in their front). We usually abbreviate $t \in T$, for a number term t and a string term T, as $T(t)$. For a language $\mathcal{L} \supseteq \mathcal{L}_A^2$ we write $\Sigma_0^B(\mathcal{L})$ to denote Σ_0^B formulas in the language $\mathcal{L}$.

Our axiomatization is intended to mimic the behaviour of these functions and relations in the two-sorted standard model $\mathbb{N}_2$. To this end we will use the following axioms, which define the theory $\mathbf{V}^0$.

Basic 1. $x + 1 \neq 0$	**Basic 2**. $x + 1 = y + 1 \rightarrow x = y$
Basic 3. $x + 0 = x$	**Basic 4**. $x + (y + 1) = (x + y) + 1$
Basic 5. $x \cdot 0 = 0$	**Basic 6**. $x \cdot (y + 1) = (x \cdot y) + x$
Basic 7. $(x \leq y \wedge y \leq x) \rightarrow x = y$	**Basic 8**. $x \leq x + y$
Basic 9. $0 \leq x$	**Basic 10**. $x \leq y \vee y \leq x$
Basic 11. $x \leq y \leftrightarrow x < y + 1$	**Basic 12**. $x \neq 0 \rightarrow \exists y \leq x(y + 1 = x)$
L1. $X(y) \rightarrow y < \lvert X\rvert$	**L2**. $y + 1 = \lvert X\rvert \rightarrow X(y)$

SE. $(|X| = |Y| \wedge \forall i < |X|\,(X(i) \leftrightarrow Y(i))) \rightarrow X = Y$

Σ_0^B**-COMP.** $\exists X < y \forall z < y(X(z) \leftrightarrow \varphi(z))$, for all $\varphi \in \Sigma_0^B$ where X does not occur free in φ.

Here, **Basic 1** through **Basic 12** define the basic properties of addition, multiplication and ordering and their interplay with the constants zero and one. **L 1** and **L 2** settle properties of the "size" of a set. As mentioned before we want to perceive a set as the graph of its characteristic function and therefore its "size" is actually referring to the length of this representation. The axiom **SE** is the axiom of Extensionality and states that two sets are equal if and only if they have the same length and contain the same elements. The Comprehension Axiom Σ_0^B-**Comp** claims the existence of all Σ_0^B-definable sets. These axioms at hand we can prove some basic properties of $\mathbf{V}^0$. See [29] for their proofs.

For a family $\Phi \subset \mathcal{L}_A^2$, we let the Φ-induction scheme be for all $\varphi \in \Phi$,

$$(\forall x(\varphi(0) \wedge \varphi(x) \rightarrow \varphi(x + 1))) \rightarrow \forall x \varphi(x).$$

We let the Φ-minimization scheme be for all $\varphi \in \Phi$,

$$\varphi(y) \rightarrow \exists x \leq y(\varphi(x) \wedge \neg\exists z < x \varphi(z)).$$

In a similar fashion we let the Φ-maximization scheme be for all $\varphi \in \Phi$,

$$\varphi(0) \rightarrow \exists x \leq y(\varphi(x) \wedge \neg\exists z \leq y(x < z \wedge \varphi(z))).$$

Proposition 1.26. $\mathbf{V}^0$ proves the Σ^B_0-induction scheme, the Σ^B_0-minimization scheme and the Σ^B_0-maximization scheme.

Also, $\mathbf{V}^0$, if restricted to its number part does not prove more than $\mathbf{I\Delta}_0$. The following theorem holds.

Proposition 1.27. $\mathbf{V}^0$ is a conservative extension of $\mathbf{I\Delta}_0$.

We can extend Parikh's Theorem to the two-sorted case. To this end we first

Theorem 1.28 (Parikh, two-sorted)**.** *Let T be a two-sorted, polynomially bounded theory extending $\mathbf{V}^0$ and $\varphi(\bar{x}, \bar{y}, \bar{X}, \bar{Y})$ be a bounded formula with all variables shown. Then, if*

$$T \vdash \forall \bar{x} \forall \bar{X} \exists \bar{y} \exists \bar{Y} \varphi(\bar{x}, \bar{y}, \bar{X}, \bar{Y}),$$

it also holds that

$$T \vdash \forall \bar{x} \forall \bar{X} \exists \bar{y} \leq t \exists \bar{Y} \leq t \varphi(\bar{x}, \bar{y}, \bar{X}, \bar{Y}),$$

where t is a bounded term containing only the variables $\bar{x}, \bar{X}$.

Definable Relations and Functions

We will now give some $\mathbf{V}^0$-definable functions and relations that we need for our arguments. For their proper definitions as well as for a thorough treatment of the notion of definability with respect to $\mathbf{V}^0$, we refer the reader to Section A.2.1 in Appendix A.

If a relation R is Σ^B_0-definable in $\mathbf{V}^0$, then so is its *characteristic function* χ_R. We can define *sequences of constant length* of numbers by a standard encoding as numbers. We will denote such sequences by $\langle x_1, \ldots, x_k \rangle$, where the x_i are the elements and k is a fixed standard number. We can also code sequences of numbers, which are not of constant length. Informally, the Σ^B_0-formula $seq(i, Z)$, defined by Formula (A.9) in Appendix A, returns the ith element of the sequence coded by Z. For brevity we will use $Z[i]$ to refer to the ith element in the sequence coded by Z. This encoding is not unique, but we can make it unique, by ensuring that we talk about the lexicographically smallest Z encoding the sequence. We will refer to this encoding by Formula (A.10) and call the formula $SEQ(i, Z)$. Thus, we can also refer to the *length of a sequence* by using a Σ^B_0-formula, which we call $length(Z)$. This also allows us to define *higher dimensional sequences* of numbers, i.e. matrices, etc. We will refer to these elements as $Z[i_1, \ldots, i_k]$ and to lower dimensional subsequences in the same way, using $\cdot$ to represent the places, where the variable is still free. For example we refer to the first row in the matrix $Z[i, j]$ by $Z[1, \cdot]$. This also gives us a means of encoding a sequence of strings, as it is essentially the same. There is a general way of extending the language $\mathcal{L}^2_A$ with new relation and function symbols, such that the theory extending $\mathbf{V}^0$ by defining axioms of the symbols in the extended language, still has comprehension and induction for the new statements and is conservative over $\mathbf{V}^0$. See Section V.4 in [29] or, for a brief review, Section A.2.1 in Appendix A.

In the following two sections we will briefly review two extensions of $\mathbf{V}^0$ that are subject to the later research. We focus only on the properties that are necessary for our results. See Appendix A or [29] for a more thorough treatment of $\mathbf{VTC}^0$ and [29] for one on $\mathbf{VNC}^k$.

1.3.1 The Theory $\mathbf{VTC}^0$

The theory $\mathbf{VTC}^0$ is an extension of $\mathbf{V}^0$ by an axiom that ensures that we are able to count the size of strings. As we are not able to prove the Pigeonhole Principle in $\mathbf{V}^0$, the theory resulting from the addition of the axiom is properly stronger. Formally we let $\mathbf{VTC}^0$ consist of the theory $\mathbf{V}^0$ together with the axiom NUMONES defined as follows:

Definition 1.29 (NUMONES)**.** Let $\delta_{\mathsf{NUM}}(y, X, Z)$ be the following Σ_0^B formula:

$$\delta_{\mathsf{NUM}}(y, X, Z) := SEQ(y, Z) \wedge Z[0] = 0 \wedge \forall u < y((X(u) \rightarrow Z[u+1] = Z[u] + 1) \wedge (\neg X(u) \rightarrow Z[u+1] = Z[u])). \tag{1.1}$$

Define NUMONES to be the following Σ_1^B formula:

$$\mathsf{NUMONES} := \exists Z \leq 1 + \langle y, y\rangle . \delta_{\mathsf{NUM}}(y, X, Z). \tag{1.2}$$

Thus, NUMONES basically guarantees the existence of a sequence which counts the number of 1's in a given string, that is, it measures the size of the bounded set related to that string.

Using NUMONES we can define the function $numones(y, X)$ that, given y and X, returns the yth entry of $Z(X)$ via the following Σ_1^B-defining axiom

$$numones(y, X) = z \leftrightarrow \exists Z \leq 1 + \langle |X|, |X| \rangle (\delta_{\mathsf{NUM}}(|X|, X, Z) \wedge Z[y] = z). \tag{1.3}$$

We shall use the following abbreviation:

$$numones(X) := numones(|X| - 1, X).$$

This axiom not only allows to count the number of elements of a set, but also gives us the means to compute sums with a moderately large number of summands, as well as the product of two large numbers. Much like for $\mathbf{V}^0$, we can also extend $\mathbf{VTC}^0$ in a conservative way by defining axioms for symbols in an extended language. The function *numones* can be introduced like this and we will make use of such introduction several times to obtain a more convenient theory for the task at hand. See Section A.2.2 in Appendix A and also Sections I.X.3.2 and I.X.3.3 in [29]. We will start out by describing how to do basic Linear Algebra in $\mathbf{VTC}^0$.

Some Elementary Linear Algebra in $\mathbf{VTC}^0$

Observe that proper Linear Algebra is well beyond the scope of TC^0 and that is even more the case when investigating the related theory $\mathbf{VTC}^0$. Therefore, what we will introduce here can only be considered as the very basics of Linear Algebra. For a more thorough treatment of Linear Algebra, we would need to strengthen the theory (see [89] for this). Using the pairing function available in $\mathbf{V}^0$, we can code a subset of the *rational numbers*, more precisely we code a number $\frac{x}{y}$ as the pair $\langle x, y\rangle$. The numbers we will be working with will all have a common denominator n^{2c}. This limits the number of multiplications we can do for each number, but this will not pose a problem in our case. We can also define

the usual operations on these numbers (see for example [69]). Given a string Z coding a sequence, we can compute the value of the sum of its elements, that is, we can Σ^B_1-define a formula $\mathsf{sum}(y, Z)$ that represents $\sum_{i=0}^{y} Z[i]$ and prove its main properties as well as its totality in $\mathbf{VTC}^0$ (see Proposition 1.30 below or Proposition A.39 in Appendix A for the proof). We define *rational vectors* in a natural way as sequences of rational numbers. Using the sum function we can define an inner product of vectors of the same length. For vectors $\vec{u}, \vec{v} \in \mathbb{Q}^k$ we denote the inner product by $\langle \vec{u}, \vec{v} \rangle$. For a rational matrix M and a rational vector $\vec{v}$, we can Σ^B_1-define their product $M\vec{v}$.

Proposition 1.30 (Basic properties of sums in $\mathbf{VTC}^0$)**.** In what follows we consider the theory $\mathbf{VTC}^0$ over an extended language (including possibly new Σ^B_1-definable function symbols in $\mathbf{VTC}^0$ and their defining axioms). The function $f(i)$ is a number function symbol mapping to the rationals or naturals (possibly with additional undisplayed parameters). The theory $\mathbf{VTC}^0$ proves the following statements:

Substitution: Assume that $u(i), v(i)$ are two terms (possibly with additional undisplayed parameters), such that $u(i) = v(i)$ for any $i \leq n$, then

$$\sum_{i=0}^{n} f(u(i)) = \sum_{i=0}^{n} f(v(i)).$$

Distributivity: Assume that u is a term that does not contain the variable i, then

$$u \cdot \sum_{i=0}^{n} f(i) = \sum_{i=0}^{n} u \cdot f(i).$$

Rearranging: Assume that $I = \{0, \ldots, n\}$ and let $I_1, \ldots, I_k$ be a definable partition of I (specifically, the sets $I_1, \ldots, I_k$ are all Σ^B_0-definable in $\mathbf{VTC}^0$ and $\mathbf{VTC}^0$ proves that the I_j's form a partition of I). Then

$$\sum_{i=0}^{n} f(i) = \sum_{j=1}^{k} \sum_{i \in I_j} f(i),$$

where $\sum_{i \in I_j} f(i)$ denotes the term $\sum_{i=0}^{|I_j|-1} f(\delta(i))$ where $\delta(i)$ is the function that enumerates (in ascending order) the elements in I_j.

Inequalities: Let $g(i)$ be a number function mapping to the rationals or naturals (possibly with additional undisplayed parameters), such that $f(i) \leq g(i)$ for all $0 \leq i \leq n$, then

$$\sum_{i=0}^{n} f(i) \leq \sum_{i=0}^{n} g(i).$$

Additionally to being able to compute the sizes of bounded sets, we can also prove some elementary counting properties. We will now give a synopsis of Section A.2.2 in Appendix A and refer to that section for the proofs.

Notation: When reasoning in the theory $\mathbf{VTC}^0$, we will say that a family of Σ^B_0-definable in $\mathbf{VTC}^0$ sets $B_0, \ldots, B_\ell$ *forms a partition of a set* B whenever

$\mathbf{VTC}^0$ proves that (i) $\bigcup_{i=0}^{\ell} B_i = B$, and (ii) $B_i \cap B_j = \emptyset$, for all $0 \leq i \neq j \leq \ell$. Here, $\bigcup_{i=0}^{\ell} B_i := \{r \ : \ \exists i \leq \ell, B_i(r)\}$.

Proposition 1.31 (Some counting in $\mathbf{VTC}^0$)**.** Let $B_1, \ldots, B_\ell$ be family of Σ_0^B-definable sets in $\mathbf{VTC}^0$ that partition the set B (ℓ may be a variable). Then, $\mathbf{VTC}^0$ proves:

$$numones(B) = \sum_{i=1}^{\ell} numones(B_i)\,.$$

Proposition 1.32 (More counting in $\mathbf{VTC}^0$)**.** Let $\varphi(x)$ be a Σ_0^B formula (possibly in an extended language of $\mathbf{VTC}^0$). The theory $\mathbf{VTC}^0$ can prove that if $Z = \{0 \leq i < m \ : \ \varphi(i)\}$ and for any $0 \leq i < m$,

$$\gamma_i = \begin{cases} a, & \varphi(i); \\ b, & \neg\varphi(i), \end{cases}$$

then

$$\sum_{i<m} \gamma_i = a \cdot numones(Z) + b \cdot (m - numones(Z)).$$

For a number term t, we write $\forall x \in [t]\, \Phi$ to abbreviate $\forall x \leq t(x \geq 1 \rightarrow \Phi)$.

Proposition 1.33. The theory $\mathbf{VTC}^0$ proves the following statement. Let $F(x)$ be a string function. Let $d < t$ be a natural number and assume that any number in any set $F(1), \ldots, F(t)$ occurs in at most d many sets in $F(1), \ldots, F(t)$. Let $g(x)$ be a number function such that $g(1), \ldots, g(t)$ are (not necessarily distinct) numbers with $g(i) \in F(i)$ for all $i \in [t]$. Then $numones(\{g(i) \ : \ i \in [t]\}) \geq \lceil t/d \rceil$.

1.3.2 The Theories $\mathbf{VNC}^k$ and VNC

As before we will axiomatize theories corresponding to the provability strength of circuit classes by adding to $\mathbf{V}^0$ an axiom that postulates the existence of a witness for a complete problem for that circuit class. In this section we will treat NC^k circuits and, following [29], we will use the layered monotone circuit value problem, which is complete for these classes, to axiomatize the corresponding theories $\mathbf{VNC}^k$. We will be more precise now.

Definition 1.34. The *monotone circuit value problem* for some circuit class $\mathcal{C}$ is, given a monotone circuit $C \in \mathcal{C}$ and an input I to C, to decide whether $C(I) = 1$ or not. The *layered monotone circuit value problem* is the same for layered circuits.

These problems are hard for their respective classes.

Proposition 1.35. Let $k \geq 0$. The layered and non-layered monotone circuit value problems for AC^k and NC^{k+1} are hard for the classes AC^k and NC^{k+1}, respectively.

We will now formalize these problems. To do so, we will first consider the case where $k = 1$. In this case any NC^k-circuit can efficiently be turned into an

equivalent formula. The circuit value problem thus reduces to a formula value problem, which is the base of the following definition of $\mathbf{VNC}^1$.

We perceive a monotone formula φ as a encoded by a binary tree G, where the leafs are labeled with the variables and each node is labeled by a connective, i.e. by either $\wedge$ or $\vee$. An assignment is a string I, that assigns to each variable a value, i.e. 0 or 1. The following formula is a formalization of the statement that there exists an evaluation Y for each such encoding of a formula:

$$MFV \equiv \exists Y \leq 2a+1.\delta_{MFV}(a,G,I,Y), \text{ where}$$

$$\begin{aligned}\delta_{MFV}(a,G,I,Y) \equiv \forall x < a((Y(x+a) \leftrightarrow I(x)) \wedge Y(0) \wedge \\ 0 < x \rightarrow (Y(x) \leftrightarrow ((G(x) \wedge Y(2x) \wedge Y(2x+1)) \vee \\ (\neg G(x) \wedge (Y(2x) \vee Y(2x+1)))))).\end{aligned}$$

Definition 1.36. We let $\mathbf{VNC}^1$ be $\mathbf{V}^0$ augmented by MFV.

To further generalize this notion we first need to find a convenient way of formalizing the notion of an AC^k and of an NC^k circuit. This is done as follows.

We define a circuit in $\mathcal{L}_A^2$ by defining the underlying set of nodes via a predicate $G[\langle a,b\rangle]$ that is true iff the bth node in layer a is $\wedge$ and false iff it is $\vee$. The nodes are connected via an edge relation $E[\langle a,b,c\rangle]$ that is true iff node b on layer a is connected with node c on layer $a+1$.

The algorithm that we want to formalize, starts in the lowest layer and computes the values of each node in this layer from the input I it then successively computes the values of all nodes in each layer from the layer below. Therefore it needs space O(size of the layer) and can be formalized as follows.

$$\begin{aligned}\delta_{LMCV}(n,d,E,G,I,Y) \equiv \forall x < n \forall z < d((Y[\langle 0,x\rangle] \leftrightarrow I[x])) \wedge \\ (Y[\langle z+1,x\rangle] \leftrightarrow \\ (G[\langle z+1,x\rangle] \wedge \forall u < n(E(z,u,x) \rightarrow Y(z,u))) \vee \\ (\neg G[\langle z+1,x\rangle] \wedge \exists u < n(E(z,u,x) \rightarrow Y(z,u))))).\end{aligned}$$

As the circuit we want to compute is an NC circuit we have to assume that the fan-in is 2. We formalize that as

$$\begin{aligned}Fanin2(n,d,E) \equiv \forall z < d \forall x < n \exists u_1 < n \exists u_2 < n \forall v < n \\ (E(z,v,x) \rightarrow (v = u_1 \vee v = u_2)).\end{aligned}$$

We are now in the position to define the theories that correspond to the circuit classes NC^k.

Definition 1.37. Let $k > 1$. Then $\mathbf{VNC}^k$ is axiomatized by $\mathbf{V}^0$ extended by the following axiom

$$Fanin2(n,|n|^k,E) \rightarrow \exists Y \leq \langle |n|^k, n\rangle \delta_{LMCV}(n,|n|^k,E,G,I,Y)$$

The theory $\mathbf{VNC}$ is defined by adding the above axiom for all k.

Obviously the theories form a hierarchy, though we do not know, whether it is strict.

Observe that none of the $\mathbf{VNC}^k$ is conservative over $\mathbf{V}^0$, as, by the Witnessing Theorem for $\mathbf{V}^0$, a Σ^B_0 definition of the satisfaction relation would be sufficient to prove that $\mathsf{NC}^1 \subseteq \mathsf{AC}^0$, at least for monotone functions, which is known to be false.

The theories $\mathbf{VNC}^k$ are at least as strong as $\mathbf{VTC}^0$. See [29] Section IX.5.4.

Theorem 1.38. $\mathbf{VNC}^1$ *proves* NUMONES. *Therefore everything provable in* $\mathbf{VTC}^0$ *is also provable in* $\mathbf{VNC}^k$ *for any* $k \geq 1$.

1.3.3 Relation between Arithmetic Theories and Proof Systems

In this section we will remind the reader of a connection between the Theory $\mathbf{V}^0$ and some of its extensions and certain propositional proof systems (see also [29][57]).

Definition 1.39. The following predicates will be subsequently used. They are definable with respect to $\mathbf{V}^0$ (see [57]).

- $\mathsf{Fla}(X)$ is a Σ^B_0 formula that says that the the string X codes a formula.
- $\mathsf{DNF}(X)$ is a Σ^B_0 formula that says that the the string X codes a formula in DNF.
- $Z \models X$ is the Δ^B_1 definable property that the truth assignment Z satisfies the formula X.
- $\mathsf{Taut}(X)$ is the Π^B_1 formula $\mathsf{Fla}(X) \wedge \forall Z \leq t(|X|)Z \models X$, where t is a number term.
- $\mathsf{Prf}_{F_d}(\Pi, A)$ is a Σ^B_0 definable predicate meaning Π is a depth d Frege proof for A.
- $\mathsf{Prf}_{\mathsf{TC}^0-F_d}(\Pi, A)$ is a Σ^B_0 definable predicate meaning Π is a depth d TC^0-Frege proof for A.
- $\mathsf{Prf}_F(\Pi, A)$ is a Σ^B_0 definable predicate meaning Π is a Frege proof for A.

We denote the depth by a formula coded by a string X by $\mathsf{dp}(X)$. This is clearly Σ^B_0 definable in $\mathbf{V}^0$. Observe that in $\mathbf{V}^0$ we cannot prove that every formula has an evaluation.

The following holds

Theorem 1.40 (see [29])**.** *The Theory* $\mathbf{V}^0$ *proves that* AC^0*-Frege is sound, i.e. for every* d

$$\forall A \forall \Pi \mathsf{Prf}_{F_d}(\Pi, A) \wedge \mathsf{dp}(A) \leq d \rightarrow \mathsf{Taut}(A).$$

Theorem 1.41 (see [29])**.** *The Theory* $\mathbf{VTC}^0$ *proves that* TC^0*-Frege is sound, i.e.*

$$\forall A \forall \Pi \mathsf{Prf}_{\mathsf{TC}^0-F_d}(\Pi, A) \wedge \mathsf{dp}(A) \leq d \rightarrow \mathsf{Taut}(A).$$

Theorem 1.42 (see [29]). *The Theory* $\mathbf{VNC}^1$ *proves that Frege is sound, i.e.*

$$\forall A \forall \Pi \mathsf{PRF}_F(\Pi, A) \to \mathsf{Taut}(A).$$

On the other hand, provability of the universal closure of Σ_0^B formulas in $\mathbf{V}^0$ and $\mathbf{VNC}^1$ implies the existence of polynomial size proofs of their propositional translations in AC^0-Frege, TC^0-Frege and Frege, respectively.

The Paris-Wilkie Translation

We will show now, how one can translate a Σ_0^B formula φ into a family of propositional formulas $[\![\varphi]\!]$ and use this translation to infer a relation between provability in $\mathbf{V}^0$ and its extensions to efficient provability in bounded depth Frege, TC^0-Frege and Frege. This is the Paris-Wilkie Translation defined in [73].

Definition 1.43 (Propositional translation $[\![\cdot]\!]$ of Σ_0^B formulas). Let $\varphi(\bar{x}, \bar{X})$ be a Σ_0^B formula. The *propositional translation* of φ is a family

$$[\![\varphi]\!] = \{[\![\varphi]\!]_{\bar{m};\bar{n}} \mid m_i, n_i \in \mathbb{N}\}$$

of propositional formulas in variables $p_j^{X_i}$ for every $X_i \in \bar{X}$. The intended meaning is that $[\![\varphi]\!]$ is a valid family of formulas if and only if the formula

$$\forall \bar{x} \forall \bar{X} \left((\bigwedge |X_i| = \underline{n_i}) \to \varphi(\underline{\bar{m}}, \bar{X}) \right)$$

is true in the standard model $\mathbb{N}_2$ of two sorted arithmetic, where $\underline{n}$ denotes the nth numeral, for any $n \in \mathbb{N}$.

For given $\bar{m}, \bar{n} \in \mathbb{N}$ we define $[\![\varphi]\!]$ by induction on the size of the formula $[\![\varphi]\!]_{\bar{m};\bar{n}}$. We denote the value of a term t by $\mathsf{val}(t)$.

Case 1: Let $\varphi(\bar{x}, \bar{X})$ be an atomic formula.

- If $\varphi(\bar{x}, \bar{X})$ is $\top$ (or $\bot$), then $[\![\varphi]\!]_{\bar{m},\bar{n}} := \top$ (or $\bot$).
- If $\varphi(\bar{x}, \bar{X})$ is $X_i = X_i$, then $[\![\varphi]\!]_{\bar{m},\bar{n}} := \top$.
- If $\varphi(\bar{x}, \bar{X})$ is $X_i = X_j$ for $i \neq j$, then (using the fact that $\mathbf{V}^0$contains the extensionality axiom **SE**) instead of translating φ, we translate the $\mathbf{V}^0$-equivalent formula

$$|X_i| = |X_j| \wedge \forall k \leq |X| \, (X_i(k) \leftrightarrow X_j(k))).$$

- If $\varphi(\bar{x}, \bar{X})$ is $t_1(\bar{y}, |\bar{Y}|) = t_2(\bar{z}, |\bar{Z}|)$ for terms t_1, t_2, number variables $\bar{y}, \bar{z}$ and string variables $\bar{Y}, \bar{Z}$, where $\bar{y} \cup \bar{z} = \bar{x}$ and $\bar{Y} \cup \bar{Z} = \bar{X}$, and $\underline{\bar{m}^y}, \underline{\bar{m}^z}$ and $\underline{\bar{n}^Y}, \underline{\bar{n}^Z}$ denote the corresponding assignments of numerals $\underline{\bar{m}}, \underline{\bar{n}}$ to the $\bar{y}, \bar{z}$ and $\bar{Y}, \bar{Z}$ variables, respectively. Then

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \begin{cases} \top & \text{if } \mathsf{val}(t_1(\underline{\bar{m}^Y}, \underline{\bar{n}^Y})) = \mathsf{val}(t_2(\underline{\bar{m}^Z}, \underline{\bar{n}^Z})) \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

- If $\varphi(\bar{x}, \bar{X})$ is $t_1(\bar{y}, |\bar{Y}|) \leq t_2(\bar{z}, |\bar{Z}|)$ for terms t_1, t_2, number variables $\bar{y}, \bar{z}$ and string variables $\bar{Y}, \bar{Z}$, then

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \begin{cases} \top & \text{if } \mathsf{val}(t_1(\underline{\bar{m}^Y}, \underline{\bar{n}^Y})) \leq \mathsf{val}(t_2(\underline{\bar{m}^Z}, \underline{\bar{n}^Z})) \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

- If $\varphi(\bar{x}, \bar{X})$ is $X_i(t(\bar{x}, |\bar{X}|))$, then

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \bot \text{ if } n_i = 0$$

and otherwise

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \begin{cases} p^{X_i}_{\mathsf{val}(t(\underline{\bar{m}},\underline{\bar{n}}))} & \text{if } \mathsf{val}(t(\underline{\bar{m}},\underline{\bar{n}})) < \underline{n_i - 1}, \\ \top & \text{if } \mathsf{val}(t(\underline{\bar{m}},\underline{\bar{n}})) = \underline{n_i - 1}, \\ \bot & \text{if } \mathsf{val}(t(\underline{\bar{m}},\underline{\bar{n}})) > \underline{n_i - 1}. \end{cases}$$

Case 2: The formula φ is not atomic.

- If $\varphi \equiv \psi_1 \wedge \psi_2$ we let

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := [\![\psi_1]\!]_{\bar{m},\bar{n}} \wedge [\![\psi_2]\!]_{\bar{m},\bar{n}}.$$

- If $\varphi \equiv \psi_1 \vee \psi_2$ we let

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := [\![\psi_1]\!]_{\bar{m},\bar{n}} \vee [\![\psi_2]\!]_{\bar{m},\bar{n}}.$$

- If $\varphi \equiv \neg\psi$ we let

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \neg[\![\psi]\!]_{\bar{m},\bar{n}}.$$

- If $\varphi \equiv \exists y \leq t(\bar{x}, |\bar{X}|)\psi(y, \bar{x}, \bar{X})$ then

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \bigvee_{i=0}^{\mathsf{val}(t(\underline{\bar{m}},\underline{\bar{n}}))} [\![\psi(\underline{i}, \bar{x}, \bar{X})]\!]_{\bar{m},\bar{n}}.$$

- If $\varphi \equiv \forall y \leq t(\bar{x}, |\bar{X}|)\psi(y, \bar{x}, \bar{X})$ then

$$[\![\varphi]\!]_{\bar{m},\bar{n}} := \bigwedge_{i=0}^{\mathsf{val}(t(\underline{\bar{m}},\underline{\bar{n}}))} [\![\psi(\underline{i}, \bar{x}, \bar{X})]\!]_{\bar{m},\bar{n}}.$$

This concludes the translation for Σ^B_0 formulas.

Proposition 1.44. There exists a polynomial p such that for all Σ^B_0 formulas $\varphi(\bar{x}, \bar{X})$ the following holds

- If $\mathbf{V}^0 \vdash \forall\bar{X}\forall\bar{x}\varphi(\bar{x}, \bar{X})$, then there exist a d such that all $[\![\varphi]\!]_{\bar{m},\bar{n}}$ have depth d Frege proofs of length at most $p(\max(\bar{m}, \bar{n}))$, for any $\bar{m}, \bar{n}$.
- If $\mathbf{VTC}^0 \vdash \forall\bar{X}\forall\bar{x}\varphi(\bar{x}, \bar{X})$, then there exist a d such that all $[\![\varphi]\!]_{\bar{m},\bar{n}}$ have depth d TC^0-Frege proofs of length at most $p(\max(\bar{m}, \bar{n}))$, for any $\bar{m}, \bar{n}$.

- If $\mathbf{VNC}^1 \vdash \forall \bar{X} \forall \bar{x} \varphi(\bar{x}, \bar{X})$, then there exist Frege proofs of all $[\![\varphi]\!]_{\bar{m},\bar{n}}$ of length at most $p(\max(\bar{m}, \bar{n}))$, for any $\bar{m}, \bar{n}$.

Propositions 1.40, 1.41 and 1.42 are examples of general principles, the so called *Reflection Principles*, which we will state now only for formulas in DNF, as $\mathbf{V}^0$ is not strong enough to evaluate general formulas.

Definition 1.45 (Reflection Principle)**.** Let P be a pps. Then the *Reflection Principle* for P, Ref_P, is the $\forall\Delta_1^B$-formula (w.r.t. $\mathbf{V}^0$)

$$\forall \Pi \forall X \forall Z((\mathsf{DNF}(X) \wedge \mathsf{Prf}_P(\Pi, X)) \rightarrow (Z \vDash X)),$$

where Prf_P is a Δ_1^B-predicate formalizing P-proofs.

Reflection Principles condense the strength of propositional proof systems. The following result exemplifies this. A detailed exposition can be found in [29], chapter X, or in [57], chapter 9.3.

Theorem 1.46. *1. If $\mathbf{V}^0 \vdash \mathsf{Ref}_F$ then bounded depth Frege simulates Frege w.r.t. DNF formulas.*

2. If $\mathbf{V}^0 \vdash \mathsf{Ref}_{\mathsf{TC}^0-F}$ then bounded depth Frege simulates TC^0-Frege w.r.t. DNF formulas.

3. If $\mathbf{VTC}^0 \vdash \mathsf{Ref}_F$ then TC^0-Frege simulates Frege w.r.t. DNF formulas.

We will only give a brief sketch of the proof of item 1. here and leave out the technical details.

Sketch. Let φ be a formula and π_φ a Frege proof of φ. Since $\mathbf{V}^0$ proves Ref_F, by Propositions 1.40 and 1.44 we have polynomial size proofs of its translations $[\![\mathsf{Ref}_F]\!]$ in bounded depth Frege. Bounded depth Frege itself, however, is strong enough to verify that a proper encoding of the computation of the Turing machine verifying the Frege proof π_φ is correct. Thus it can verify that π_φ is a Frege-proof and, using the translation of the Reflection Principle and the Cut rule, conclude $[\![\mathsf{Taut}(\varphi)]\!]$. From this φ follows, cf. [57] Lemma 9.3.7. □

As an application of Theorem 1.46 we can state the following observation.

Proposition 1.47. $\mathbf{VTC}^0$ is not a conservative extension of $\mathbf{V}^0$.

Proof. This statement follows from Theorem 1.9 and Theorem 1.14. If $\mathbf{V}^0$ could prove the same statements as $\mathbf{VTC}^0$, then, by Proposition 1.41, it could especially prove the Reflection Principle for TC^0-Frege. This, in conjunction with Theorem 1.14 and Proposition 1.44 yields a polynomially bounded, bounded depth Frege proof of the Pigeonhole Principle. That, however, is a contradiction to Theorem 1.9. □

2. Refutations of Random 3CNF in TC^0-Frege

In this chapter we will give a synopsis of [67], the whole article is given as Appendix A in the end of the thesis.

By Theorem 1.5 we know that Resolution fails to have efficient proofs of the unsatisfiability of random 3CNF, if the clause-to-variable ratio is below $n^{0,5-\epsilon}$, where n is the number of variables. This is contrasted by Theorem 1.4, which states that these statements are already well beyond the unsatisfiability threshold of $8 \cdot \ln(2)$ clauses per variable, which suggests that proofs of unsatisfiability should be easier to do. This suggestion is justified. In [37], Feige, Kim and Ofek showed that there is a polynomially verifiable witness of the unsatisfiability of 3CNF, if the clause-to-variable ratio is at least $c \cdot n^{0,4}$ for some constant c. This is our starting point. We will show that this witness can be understood with the means of TC^0-Frege and then conclude that 3CNF with more than $c \cdot n^{0,4}$ clauses per variable can be efficiently refuted by TC^0-Frege proofs (by which we mean that TC^0-Frege proves their negation efficiently). Our main result is the following:

Theorem 2.1. *With probability $1-o(1)$ a random 3CNF formula with n variables and $cn^{1.4}$ clauses (for a sufficiently large constant c) has polynomial-size TC^0-Frege refutations.*

We will now give an outline of the structure of the proof of Theorem 2.1. Instead of directly constructing TC^0-Frege proofs we will work in $\mathbf{VTC}^0$ and use the relation between this theory and the proof system, as mentioned in Section 1.3.3. We have seen there that, when restricted to proving Σ_0^B statements, the theory $\mathbf{VTC}^0$ characterizes uniform polynomial-size TC^0-Frege proofs. The construction of polynomial-size TC^0-Frege refutations for random 3CNF formulas, will consist of the following steps:

I. Formalize the following statement as an $\mathcal{L}_A^2$ formula:

$$\forall \text{ assignment } A \ \big(\mathbf{C} \text{ is a 3CNF and } w \text{ is its FKO unsatisfiabiliy witness} \longrightarrow \text{exists a clause } C_i \text{ in } \mathbf{C} \text{ such that } C_i(A) = 0\big), \tag{2.1}$$

where an *FKO witness* is a suitable formalization of the unsatisfiability witness defined by Feige, Kim and Ofek [37]. We will call the corresponding predicate the *FKO predicate*.

II. Prove formula (2.1) in the theory $\mathbf{VTC}^0$.

III. Translate the proof in Step **II.** into a family of propositional TC^0-Frege proofs (of the family of propositional translations of (2.1)). By Proposition 1.44, this will be a polynomial-size propositional proof (in the size of $\mathbf{C}$). The translation of (2.1) will consist of a family of propositional formulas of the form:

$$[\![\mathbf{C} \text{ is a 3CNF and } w \text{ is its FKO unsatisfiabiliy witness}]\!] \longrightarrow [\![\text{exists a clause } C_i \text{ in } \mathbf{C} \text{ such that } C_i(A) = 0]\!]. \tag{2.2}$$

By the nature of the propositional translation, (second-sort) variables in the original first-order formula translate into a collection of propositional variables. Thus, (2.2) will consist of propositional variables derived from the variables in (2.1).

IV. For the next step we first notice the following two facts:

(i) Assume that $\underline{\mathbf{C}}$ is a random 3CNF with n variables and $cn^{1.4}$ clauses (for a sufficiently large constant c). By [37], with high probability there exists an FKO unsatisfiability witness $\underline{w}$ for $\underline{\mathbf{C}}$. Both $\underline{w}$ and $\underline{\mathbf{C}}$ can be encoded as finite sets of numbers, as required by the predicate for 3CNF and the FKO predicate in (2.1). Let us identify $\underline{w}$ and $\underline{\mathbf{C}}$ with their encodings. Then, assuming (2.1) was formalized correctly, assigning $\underline{w}$ and $\underline{\mathbf{C}}$ to (2.1) satisfies the *premise* of the implication in (2.1).

(ii) Now, by the definition of the translation from first-order formulas to propositional formulas, if an object α satisfies the predicate $P(X)$ (i.e., $P(\alpha)$ is true in the standard model), then there is a propositional assignment of $0, 1$ values that satisfies the propositional translation of $P(X)$. Thus, by Item (i) above, there exists an $0, 1$ assignment ζ that satisfies the premise of (2.2) (i.e., the propositional translation of the premise of the implication in (2.1)).

In the current step we show that after assigning ζ to the conclusion of (2.2) (i.e., to the propositional translation of the conclusion in (2.1)) one obtains precisely $\neg\underline{\mathbf{C}}$ (formally, a renaming of $\neg\underline{\mathbf{C}}$, where $\neg\underline{\mathbf{C}}$ is the 3DNF obtained by negating $\underline{\mathbf{C}}$ and using the de Morgan laws).

V. Take the propositional proof obtained in **III.**, and apply the assignment ζ to it. The proof then becomes a polynomial-size TC^0-Frege proof of a formula $\phi \rightarrow \neg\underline{\mathbf{C}}$, where ϕ is a propositional sentence (without variables) logically equivalent to TRUE (because ζ satisfies it, by **IV.**) From this, one can easily obtain a polynomial-size TC^0-Frege refutation of $\underline{\mathbf{C}}$ (or equivalently, a proof of $\neg\underline{\mathbf{C}}$).

The bulk of our work lies in **I.** and especially in **II.** We need to formalize the necessary properties used in proving the correctness of the FKO witnesses and show that the correctness argument can be carried out in the weak theory. There are two main obstacles in this process. The first obstacle is that the correctness (soundness) of the witness is originally proved using spectral methods, which assumes that eigenvalues and eigenvectors are over the *reals*; whereas the reals are not defined in our weak theory. The second obstacle is that one needs to prove the correctness of the witness, and in particular the part related to the spectral method, *constructively* (formally in our case, inside $\mathbf{VTC}^0$). Specifically, linear algebra is not known to be (computationally) in TC^0, and (proof-complexity-wise) it is conjectured that TC^0-Frege do not admit short proofs of the statements of linear algebra (more specifically still, short proofs relating to inverse matrices and the determinant properties; see [89] on this).

The first obstacle is solved using rational approximations of sufficient accuracy (polynomially small errors), and showing how to carry out the proof in the theory with such approximations. The second obstacle is solved basically by constructing the argument (the main formula above) in a way that exploits nondeterminism (i.e., in a way that enables supplying additional witnesses for the properties needed to prove the correctness of the original witness; e.g, (approximations of) all eigenvectors and all eigenvalues of the appropriate matrices in the original witness). In other words, we do not have to construct certain objects but can provide them, given the possibility to certify the property we need. Formally, this means that we put additional witnesses in the FKO predicate occurring in the main formula in **I.** above. This, of course, leads to TC^0-proofs that are not totally uniform.

We will now elaborate a bit more on Steps **I.** and **II.**

2.0.4 Step I.

To understand how to formalize the notion of an FKO witness, we first have to define what that notion actually is. In [37] the following was observed

1. Any satisfying assignment for a random 3CNF $\mathbf{C}$ satisfies many clauses of $\mathbf{C}$ as $3XOR$, that is, it either satisfies all or exactly one of the literals in these clauses.

2. Not too many of the clauses of $\mathbf{C}$ can be satisfied as NAE (Not All Equal), that is there are a reasonable amount of clauses, where all 3 literals are satisfied.

3. For any random 3CNF, there exist many distinct, but not disjoint sets of their clauses, such that each of this sets one of its clauses is not satisfied as $3XOR$.

If there exists too many of the sets guaranteed by item 3, we cannot satisfy item 1 anymore, so the 3CNF is not satisfiable. We will make this observation more precise now.

We call the collection of clauses in item 3) an *even k-tuple* and define it as follows

Definition 2.2 (Even k-tuple)**.** For any given k, a sequence S of k many clauses is an *even k-tuple* iff every variable appears an even number of times in the sequence. Formally, this predicate is denoted by $\mathrm{TPL}(S, k)$.

Observe that if S is an even k-tuple then k is even (since the total number of variable occurrences n is even, by assumption that each variable occurs an even number of times; and $k = n/3$, since each clause has three variables). In light of the fact that many clauses have to be satisfied as $3XOR$ (item 1. in the above observation), we define the following:

Definition 2.3 (Inconsistent k-tuple)**.** An even k-tuple is said to be *inconsistent* if the total number of negations in its clauses is odd. Formally, the predicate is denoted by $\mathrm{ITPL}(S, k)$.

Observe that at least one of the clauses in an inconsistent k-tuple cannot be satisfied as $3XOR$. Additionally, we need the the largest eigenvalue λ of the following matrix, because it provides an upper bound to the number of clauses that can be satisfied as NAE (item 2 above).

Definition 2.4 ($\textsc{Mat}(M, \mathbf{C})$)**.** We define the predicate $\textsc{Mat}(M, \mathbf{C})$ that holds iff M is an $n \times n$ rational matrix such that M_{ij} equals $\frac{1}{2}$ times the number of clauses in $\mathbf{C}$ where x_i and x_j appear with different polarity minus $\frac{1}{2}$ times the number of clauses where they appear with the same polarity. More formally, we have

$$M_{ij} := \sum_{k=0}^{m-1} E_{ij}^{(k)}, \qquad \text{for any } i, j \in [n], \tag{2.3}$$

where $E_{ij}^{(k)}$ corresponds to the kth clause in $\mathbf{C}$ as follows:

$$E_{ij}^{(k)} := \begin{cases} \frac{1}{2}, & x_i^{\varepsilon_i}, x_j^{\varepsilon_j} \in \mathbf{C}[k] \text{ and } \varepsilon_i \neq \varepsilon_j, \text{ for some } \varepsilon_i, \varepsilon_j \in \{0,1\} \text{ and } i \neq j; \\ -\frac{1}{2}, & x_i^{\varepsilon_i}, x_j^{\varepsilon_j} \in \mathbf{C}[k] \text{ and } \varepsilon_i = \varepsilon_j, \text{ for some } \varepsilon_i, \varepsilon_j \in \{0,1\} \text{ and } i \neq j; \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

We refer to Lemma 2.11 for the actual bound of the formulas satisfied as NAE. Also, the eigenvalue λ poses a problem as on the one-hand, the eigenvalue is a real number and on the other hand, we cannot construct eigenvalues in our theory, even if they are rational. To circumvent this, we define a predicate $\textsc{EigValBound}(M, \bar{\lambda}, V)$ that ensures that $\bar{\lambda}$ is a collection of n rational approximations of the eigenvalues of the matrix M and that V is the rational matrix whose rows are the rational approximations of the eigenvectors of M (where the ith row in V is the approximation of the approximate eigenvector λ_i).

The notion of imbalance provides us with a natural upper bound for the total number of satisfied literals.

Definition 2.5 (The imbalance $\textsc{Imb}(\mathbf{C}, y)$)**.** For a 3CNF $\mathbf{C}$ we define the function *i-imbalance* $\mathsf{ilmb}(\mathbf{C}, i)$ to be the absolute value of the difference of negated occurrences of x_i and non-negated occurrences of x_i in the 3CNF $\mathbf{C}$ (where $x_1, \ldots, x_n$ are considered to be all the variables in $\mathbf{C}$). It is denoted by $\mathsf{ilmb}(\mathbf{C}, i)$. For a 3CNF $\mathbf{C}$, the predicate *imbalance of* $\mathbf{C}$, denoted $\textsc{Imb}(\mathbf{C}, y)$, is true iff y equals the sum over the i-imbalances of all the variables, that is:

$$\textsc{Imb}(S, y) \leftrightarrow y = \sum_{i=1}^{n} \mathsf{ilmb}(\mathbf{C}, i).$$

We can now formulate the witness and give a sketch of the proof of its correctness. But first, we need one more definition, which gives a bound to the number of inconsistent k-tuples (as for item 3):

Definition 2.6 ((t,k,d)-collection)**.** A (t,k,d)*-collection* $\mathscr{D}$ of a 3CNF $\mathbf{C}$ with m clauses is an array of t many inconsistent k-tuples, which contain only clauses from $\mathbf{C}$, and each clause appears in at most d many such inconsistent k-tuples. The predicate is denoted $\textsc{Coll}(t, k, d, \mathbf{C}, \mathscr{D})$.

The witness for unsatisfiability of $\mathbf{C}$ contains

- The imbalance I given as $\textsc{Imb}(\mathbf{C}, y)$,
- the largest eigenvalue λ from $\textsc{Mat}(M, \mathbf{C})$ and
- a (t, k, d)-collection $\textsc{Coll}(t, k, d, \mathbf{C}, \mathscr{D})$.

Before we will state the main formula, we will restate the result from [37] of how the witness is applied.

Theorem 2.7 ([37])**.** *Let* $\mathbf{C}$ *be a 3CNF with* n *variables and* m *clauses and the FKO witness as above. If* $t > \frac{d\cdot(I+\lambda n)}{2}$*, then* $\mathbf{C}$ *is not satisfiable.*

We can now present the main formula that we are going to prove in $\mathbf{VTC}^0$. It says that if the Feige-Kim-Ofek witness fulfills the inequality $t > \frac{d\cdot(I+\lambda n)}{2} + o(1)$ (the $o(1)$ stems from the approximations) then there exists a clause in $\mathbf{C}$ that is not satisfied by any assignment A (the predicate $\textsc{NotSAT}(\mathbf{C}[i], A)$ is a straightforward formalization of this property):

Definition 2.8 (The main formula)**.** The *main formula* is the following formula ($\bar{\lambda}$ denotes n distinct number parameters $\lambda_1, \ldots, \lambda_n$):

$$\Big(\mathrm{3CNF}(\mathbf{C}, n, m) \wedge \textsc{Coll}(t, k, d, \mathbf{C}, \mathscr{D}) \wedge \textsc{Imb}(\mathbf{C}, I) \wedge \textsc{Mat}(M, \mathbf{C}) \wedge$$
$$\textsc{EigValBound}(M, \bar{\lambda}, V) \wedge \lambda = \max\{\lambda_1, \ldots, \lambda_n\} \wedge t > \frac{d\cdot(I+\lambda n)}{2} + o(1)\Big)$$
$$\longrightarrow \exists i < m\, \textsc{NotSAT}(\mathbf{C}[i], A).$$

2.0.5 Step II.

The proof of the main formula in $\mathbf{VTC}^0$ is rather tedious, so we will only give the main results needed to follow it and refer to the appendix for the details.

Theorem 2.9 (Main)**.** *The theory* $\mathbf{VTC}^0$ *proves the main formula (Definition 2.8).*

We will give a very brief sketch of the argument. The proof uses the following lemmas, the proofs of which can be found in Appendix A:

Let $\mathsf{satNAE}(A, \mathbf{C})$ be the string function that returns the set of all clauses in $\mathbf{C}$ that are satisfied as NAE by A and $\mathsf{satLit}(A, \mathbf{C})$ the set of all literals that are satisfied by A.

Lemma 2.10 (Lemma A.50 in Appendix A)**.** *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves:*

$$\mathit{numones}(\mathsf{satLit}(A, \mathbf{C})) \leq \frac{3m + I}{2}.$$

Lemma 2.11 (Lemma A.55 in Appendix A)**.** *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves:*

$$numones(\mathsf{satNAE}(A, \mathbf{C})) \leq (n\lambda + 3m)/4 + o(1).$$

Lemma 2.12 (Lemma A.56 in Appendix A)**.** *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves that the number of clauses in* $\mathbf{C}$ *that are not satisfied as 3XOR by* A *is at least* $\lceil t/d \rceil$*.*

Proof of Theorem 2.9. We will argue in $\mathbf{VTC}^0$. Assume that the premise of the main formula (see Definition 2.8) holds. By Lemma 2.10 the maximal number of literals satisfied by the assignment A is

$$\frac{3m + I}{2}.$$

By Lemma 2.11 at most

$$\frac{n\lambda + 3m}{4} + o(1)$$

clauses are satisfied as NAE. The remaining $(m - \frac{n\lambda+3m}{4} + o(1))$ clauses must be satisfied 3 times. Thus the number of clauses satisfied twice is at most

$$\frac{3m + I}{2} - 3 \cdot \underbrace{(m - \frac{n\lambda + 3m}{4})}_{satisfied\ 3\ times} - \underbrace{\frac{n\lambda + 3m}{4}}_{satisfied\ once} + o(1) = \frac{-3m + I}{2} + \frac{n\lambda + 3m}{2} + o(1)$$

$$= \frac{n\lambda + I}{2} + o(1).$$

On the other hand, by Lemma 2.12, this number is at least $\lceil t/d \rceil$. Thus we get

$$t \leq d \cdot \lceil t/d \rceil \leq \frac{d \cdot (n\lambda + I)}{2} + o(1),$$

contradicting the premise

$$t > \frac{d \cdot (I + \lambda n)}{2} + o(1).$$

□

3. Cuts of Models of $\mathbf{V}^0$

In this chapter we will give a synopsis of [66]. The full version including the proofs is given as Appendix B in the end of this thesis.

In Section 3.1, we will reproduce the main argument of the article and conclude a simulation result by exploiting an algorithm for evaluating formulas. We will give a strengthened simulation result in Section 3.2, which builds on a different algorithm that evaluates general monotone circuits. The main result of [66] is the proof of a formalized version of Nepomnjascij's Theorem [68] in a small cut of a model of $\mathbf{V}^0$. The theorem is as follows:

Theorem 3.1 (Nepomnjascij [68])**.** *Let $c \in \mathbb{N}$ and $0 < \epsilon < 1$ be constants. Then if the language $L \in \mathsf{TimeSpace}(n^c, n^\epsilon)$, the relation $x \in L$ is definable by a Σ_0^B-formula over $\mathbb{N}$.*

Using a standard evaluation algorithm the formalized version of this theorem guarantees the Σ_0^B-definability of MFV in the cut. This guarantees the existence of an evaluation, i.e. it shows that the cut is a model of $\mathbf{VNC}^1$. In Section 3.2 we will strengthen this latter result in a straightforward way to obtain a model of $\mathbf{VNC}$. In both cases that implies a subexponential simulation result with respect to the proof systems related to the theories. We will now introduce the notion of a cut $\mathcal{I}$ of a given two-sorted arithmetic model $\mathcal{M}$.

Definition 3.2 (Cut)**.** Let T be a two-sorted arithmetic theory and

$$N = \{N_1, N_2, +^N, \cdot^N, \leq^N, 0^N, 1^N, |\cdot|^N, =_1^N, =_2^N, \in^N\}$$

a model of T. A *cut*

$$M = \{M_1, M_2, +^M, \cdot^M, \leq^M, 0^M, 1^M, |\cdot|^M, =_1^M, =_2^M, \in^M\}$$

in N is any substructure such that

- $M_1 \subseteq N_1$, $M_2 \subseteq N_2$,
- $0^M = 0^N$, $1^M = 1^N$,
- M_1 is closed under $+^N, \cdot^N$ and downwards with respect to $\leq^N$,
- $M_2 = \{X \in N_2 \mid X \subseteq M_1\}$, and
- $\circ^M$ is the restriction of $\circ^N$ to M_1 and M_2 for all relation and function symbols $\circ \in \mathcal{L}_A^2$.

We call this cut the *Polylogarithmic Cut* iff

$$x \in M_1 \Leftrightarrow \exists a \in N_1, k \in \mathbb{N}\ x \leq |a|^k .$$

3.1 Polylogarithmic Cuts and VNC[1]

To infer the simulation result, we need a bounded version of the Reflection Principles we have defined in Section 1.3.3.

Given a term t and a variable x, we can also introduce the t-bounded version of the Reflection Principle for some given pps P, $\mathsf{Ref}_P(t(x))$ that claims soundness only for t-bounded proofs.

Definition 3.3 (Bounded Reflection)**.** Let t be a $\mathcal{L}^2_{Ar}$-term, x a first-sort variable and P a pps. Then the *Bounded Reflection Principle* $\mathsf{Ref}_P(t(x))$ is the formula

$$\forall \Pi \leq t(x) \forall X \leq t(x) \forall Z \leq t(x)((\mathsf{Fla}(X) \wedge \mathsf{Prf}_P(\Pi, X)) \rightarrow (Z \vDash X)).$$

We can now generalize Theorem 1.46 in the following way.

Theorem 3.4. *Let t be a $\mathcal{L}^2_A$-term and m a number variable. If $t(m) < m$ for m large enough and if $\mathbf{V}^0 \vdash \forall x \mathsf{Ref}_F(t(x))$ then for every propositional formula φ with a Frege proof of length $t(m)$ there is a bounded depth Frege proof of φ of length $m^{O(1)}$.*

The following corollary will be of interest to us.

Corollary 3.5. If $\mathbf{V}^0 \vdash \mathsf{Ref}_F(|x|^k)$ for all $k \in \mathbb{N}$, then bounded depth Frege sub exponentially simulates Frege: For all $D > 1, \delta > 0$ exists $d \geq D$, such that the existence of a Frege proof of length n of a depth D formula implies the existence of a depth d Frege proof of length at most 2^{n^δ}.

Given a polynomially bounded Turing machine A in a binary encoding, we can Σ^B_1 define a predicate $\mathsf{ACC}_A(X)$, that states that X is accepted by A. This can readily be observed, since, provided some machine A, there is a constant number of states $\sigma_1, \ldots, \sigma_k$ and the whole computation can be written into a matrix W of polynomial size. We can also define a Σ^B_1-predicate $\mathsf{REACH}_A(X, Y)$ that says that A reaches configuration Y from configuration X in at most $p(|X|)$ steps. The definition is essentially the same as for ACC. The formalized version of Nepomnjascij's Theorem now says that $\mathsf{ACC}_A(X)$ can be Σ^B_0-defined, if the machine A computes a language $L \in \mathbf{TimeSpace}(n^c, n^\epsilon)$ and the objects are small (i.e. lie in the polylogarithmic cut).

Theorem 3.6. *Let $N \vDash \mathbf{V}^0$. Let $m = |a|$ for some $a \in N_1$ and let $c, k \in \mathbb{N}$ and $\epsilon < 1$. If $L \in \mathsf{TimeSpace}(m^c, m^\epsilon)$ is computed by Turing machine A, then there exists a Σ^B_0 definition in N of the Σ^B_1-predicate ACC_A on the interval $[0, m^k]$. I.e. any $Y \in L$, bounded by m^k is Σ^B_0-definable in N and therefore exists in the polylogarithmic cut of N.*

Proof Sketch. We inductively on d define a Σ^B_0 relation $\mathsf{reach}^d_A(I, p_1, p_2, cell, \mathsf{comp})$ that states that the p_2th cell of the work tape of A, starting on configuration I and computing for $p_1 \cdot m^{d\frac{1-\epsilon}{k}}$ steps via the computation comp is *cell*. As d depends only on A and k we will be doing this induction outside of the theory to construct d many formulas. We will then prove the above mentioned properties of reach_A by Σ^B_0 induction on p_1.

It is then straightforward to prove by induction on the number of lines in comp that comp is uniquely defined by reach^0_A. We let

$$\mathsf{Reach}^0_A(I, p_1, p_2, cell) =_{def} \exists \mathsf{comp} < q(|I|)\ \mathsf{reach}^0_A(I, p_1, p_2, cell, \mathsf{comp}),$$

where q is some polynomial depending on the encoding.

Now proceed by inductively defining reach^d_A and Reach^d_A. Again, we can prove uniqueness of the computation by induction on the number of its lines and let

$$\mathsf{Reach}^d_A(i, p_1, p_2, cell) =_{def} \exists \mathsf{comp} < q(|I|)\ \mathsf{reach}^d_A(i, p_1, p_2, cell, \mathsf{comp}).$$

We now can give a Σ^B_0 definition of the predicate $W[\langle i, j, \cdot\rangle]$ coding the computation as in ACC_A on input X of length m^k. Informally the predicate $W[\langle i, j, \cdot\rangle]$ says that we compute the configurations of A by iteratively using the predicates Reach^d_A through Reach^0_A. That is, we first exhaust applications of Reach^d_A (i.e. we first make the biggest steps) until another application would lead us beyond the configuration *con* we want to check. We can then use the configuration con_d, that we obtained from the applications of Reach^d_A, in Reach^{d-1}_A to get configuration con_{d-1} and so on until we reach configuration *con*. It remains to check that $W[\langle i, j, \cdot\rangle]$ coincides with the predicate ACC_A. This follows by induction on the length of the computation. □

We can now conclude that such a cut is a model of $\mathbf{VNC}^1$ by providing an algorithm for formula evaluation. The algorithm can be found in the proof of Theorem B.14 in Appendix B.

Theorem 3.7. *Let $N \vDash \mathbf{V}^0$ and $M \subseteq N$ be the polylogarithmic cut. Then $M \vDash \mathbf{VNC}^1$.*

Using the corollary to Theorem 3.4, this yields the following simulation result first proved by Filmus, Pitassi and Santhanam [39].

Theorem 3.8 ([39])**.** *Every Frege system is sub exponentially simulated by AC^0-Frege systems.*

As $\mathbf{VNC}^1$ extends $\mathbf{VTC}^0$, we can also infer the following result about bounded depth Frege proofs of random 3CNF using Theorem 2.1 in Chapter 2.

Theorem 3.9. *For almost every random 3CNF A with n variables and $m = c{\cdot}n^{1,4}$ clauses, where c is a large constant, $\neg A$ has subexponentially bounded AC^0-Frege proofs.*

This yields a weak separation result on random instances between Resolution and bounded depth Frege.

3.2 Polylogarithmic Cuts and VNC

In this section we will generalize Theorem 3.7 in the sense that we show that the polylogarithmic cut of any $\mathbf{V}^0$ model is actually a model of $\mathbf{VNC}$ instead of only $\mathbf{VNC}^1$. This result stems from discussions with Antonina Kolokolova in the SAS programme in Cambridge and subsequently in Prague.

As before we formalize an NC^k circuit in $\mathcal{L}_A^2$ by defining the underlying set of nodes via a predicate $G[\langle a, b\rangle]$ that is true iff the bth node in layer a is $\wedge$ and false iff it is $\vee$ and an edge relation $E[\langle a, b, c\rangle]$ that is true iff node b on layer a is connected with node c on layer $a + 1$.

Keep in mind that the definition of VNC^k is

$$Fanin2(n, |n|^k, E) \rightarrow \exists Y \leq \langle |n|^k, n\rangle \delta_{LMCV}(n, |n|^k, E, G, I, Y),$$

where $Fanin2$ is the formula stating that every node has at most 2 incoming edges and $\delta_{LMCV}(n, d, E, G, I, Y)$ is the formula that states that Y is an evaluation of the layered circuit G, E, with width n and depth d, on input I. See Definition 1.37 in Section 1.3.2.

The following is a recursive algorithm computing the value of $Y[\langle j, i\rangle]$, given a circuit G, E, an input I and a node i in layer j.

```
NodeValue(G,E,I,i,j)
```

- `boolean left_value; boolean right_value;`
- `int left_succ; int right_succ;`
- `If j=0`
  - `Output I[i]; End;`
- `Else If j > 0`
  - `If G[⟨j,i⟩]=1`
    - `While k<|G⟨j-1,·⟩| AND left_succ = 0`
      - `k++;`
      - `If E(j-1,k,i)`
      - `left_succ := k;End(If);`
      - `End(While);`
    - `left_value := NodeValue(G,E,I,left_succ,j-1);`
    - `While k≥left_succ AND k<|G⟨j-1,·⟩| AND right_succ = 0`
      - `k++;`
      - `If E(j-1,k,i)`
      - `right_succ := k;End(If);`
      - `End(While);`
    - `right_value := NodeValue(G,E,I,right_succ,j-1);`
    - `Output (left_value AND right_value); End;`
  - `Else If G[⟨j,i⟩]=0`
    - `While k<|G⟨j-1,·⟩| AND left_succ = 0`
      - `k++;`
      - `If E(j-1,k,i)`
      - `left_succ := k;End(If);`
      - `End(While);`
    - `left_value := NodeValue(G,E,I,left_succ,j-1);`
    - `While k≥left_succ AND k<|G⟨j-1,·⟩| AND right_succ = 0`

```
            · k++;
            · If E(j-1,k,i)
            · right_succ := k;End(If);
            · End(While);
        * right_value := NodeValue(G,E,I,right_succ,j-1);
        * Output (left_value OR right_value); End;
```

- `Else`
  - `Output (0); End;`

Informally, the algorithm evaluates each node in a depth-first search fashion. Therefore, unlike in the algorithm anticipated in the definition of $\mathbf{VNC}^k$, it suffices to use a polylogarithmic amount of space. The algorithm clearly runs in polynomial time. It is easy to see by induction on the depth of the layer to be evaluated, that the algorithm computes the evaluation Y from the formula δ_{LMCV}. Therefore we have the following theorem.

Theorem 3.10. *If $N \models \mathbf{V}^0$ and M is its polylogarithmic cut, then $M \models \mathbf{VNC}$.*

Proof. Apply Theorem B.13 to the above algorithm. It remains to be verified that the predicate defined in this way coincides with the one from the definition of **VNC**. We will prove this via induction on the depth of the layer in the circuit. To this end let Y be the predicate whose existence is guaranteed by the definition of **VNC** and let Y' be the predicate that stems from the above algorithm.

By induction on j we will show that for every i, $Y[\langle i, j\rangle] = Y'[\langle i, j\rangle]$. If j is 0, then $Y[\langle i, j\rangle] = Y'[\langle i, j\rangle]$ because both coincide with the ith input bit. Now, to show that $Y[\langle i, j+1\rangle] = Y'[\langle i, j+1\rangle]$, we know by assumption that both matrices are the same for the two bits, which determine $Y[\langle i, j+1\rangle]$ and $Y'[\langle i, j+1\rangle]$. But the bit in Y is defined from these two in the same way as the one in Y'. Thus, they are the same. □

4. Computations in $\mathbf{VTC}^0$

In this chapter we want to shed light on the motivation that lead to [66] and the reasons why these ideas do not seem to work, if they are approached as we did. The main result we were interested in, was the alleged non-automatizability of Resolution, more precisely, we wanted to give a proof of its non-automatizability with respect to cryptographic assumptions, as was first done in [61] for EF and subsequently in [18] and [17] for F and TC^0-Frege. To use this approach in a straightforward way we would have had to prove that Resolution does not have feasible interpolation, which is not true (see [58]). However, in [6] it was shown that it is sufficient to prove that $Res(\log)$, an extension of Resolution, does not have feasible interpolation in quasipolynomial time. A question that is still open. So the main idea was to start with a theory which correspond to $Res(\log)$, which should be T^2_1, and hope to prove the correctness of RSA or the Diffie-Hellman key exchange protocol (DH) in a small cut of models of that theory. It became obvious that RSA could not play the part, as we would be needed to do exponentiation with respect to different bases. DH, on the other hand, only needed computations with respect to a fixed generator, something that could, in principle, be done using repeated squaring and a big table to look up the values. So the things that we would have to do is to assess the strength of the theory of small cuts of models of T^2_1 and then prove that this theory is strong enough to prove iterated multiplication (which we need for the repeated squaring). To start out, we first wanted to formalize in $\mathbf{VTC}^0$ iterated multiplication, as was done for TC^0 in [51]. The next step would have been to show that the cut is a model of $\mathbf{VTC}^0$ (or its one sorted equivalent). Unfortunately, it seems that $\mathbf{VTC}^0$ might be far stronger a theory than we are able to show in such cuts. The proof we had in mind was something in the lines of that of [66], that is, proving Nepomnjascij's Theorem or using a similar argument as in its proof. However, if one examines the proof of the formalized Nepomnjascij Theorem it becomes evident that the quantifier complexity is not only related to the time the algorithm needs, but also to the size of the set we wish to compute. As the sets in question have size at most $\log(a)^k$ for arbitrary k, we cannot produce a fixed quantifier-depth we need induction for. Therefore a theory such as T^2_1 is far to weak to support that argument. Another problem is, that it is seemingly very hard to carry over the results from [51] to $\mathbf{VTC}^0$. That is, even if the idea is correct, our approach to proving it does not work.

Nonetheless, understanding the computational strength of $\mathbf{VTC}^0$ is of great interest. Not in the least because a question we have to deal with eventually is, whether the approach taken in [29], to produce theories representing certain complexity classes leads to the right results. The general approach in [29] is to take a complete problem for some complexity class and add an axiom formalizing this to a theory, say $\mathbf{V}^0$. The class we get is a sort of minimal class for that complexity class. But can we really capture the computational strength in this way? It seems that the notion of completeness is too strong, as the reduction often cannot be done within the theory. So what remains is a theory representing some aspects of a complexity class and there is little reason to believe that adding a finite number of additional axioms can settle this question. But in any

case theories like $\mathbf{VTC}^0$ give us an excellent insight into the differences between theories and the real world.

To this end, here we present some partial results. We will construct an extension of $\mathbf{VTC}^0$ that proves the correctness of DH. The obvious continuation would be to get rid of the additional axioms. So far, we have not been able to do so.

4.1 DH and extensions of $\mathbf{VTC}^0$

A first observation we already mentioned before, is that the extensions of $\mathbf{V}^0$ we work with are *not* perfect representatives of the computational strength of the complexity class they are related to. This becomes apparent when e.g. trying to deduce some version of iterated multiplication including some of its properties. This all is contrasted by a result by Hesse, Allender and Barrington [51], that puts iterated multiplication into very uniform TC^0. Another observation is that iterated multiplication is possible in polylogarithmic cuts of models of $\mathbf{V}^0$, as one can easily produce a space efficient algorithm for it and then apply the formalized Nepomnjascij's Theorem 3.6. Nevertheless, we will try to formalize it properly in some theory that is given syntactically. To this end we will start with $\mathbf{VTC}^0$ and add what seems to be necessary. We will stay the whole time within the limits of TC^0-computable concepts.

For a possible application towards non-automatizability, we wish to verify in $\mathbf{VTC}^0$ the security of the *Diffie-Hellman Key Exchange Protocol (DH)* [34]. We will first sketch a proof of its correctness (with respect to the least significant bit) outside any theory to pinpoint the properties needed to be proven in $\mathbf{VTC}^0$ and then continue to work out these properties within the theory.

Definition 4.1 (Diffie-Hellman [34])**.** The *Diffie-Hellman Key Exchange Protocol (DH)*, maybe better *Diffie-Hellman-Merkle Key Exchange Protocol*, is working as follows. Assume we have two parties, Alice and Bob, who want to generate a common secret key to start a symmetrically encoded conversation. They use a cyclic group, say $\mathbb{Z}_P^*$, and a generator $g \in \mathbb{Z}_P^*$, which is supposedly public information. They then proceed as follows:

1. Alice generates randomly a number $a < P$ and sends g^a to Bob.
2. Bob generates randomly a number $b < P$ and sends g^b to Alice.
3. Alice computes $(g^b)^a$ and Bob computes $(g^a)^b$.

It is well known that DH is secure iff every bit is secure. We will now prove the correctness of the protocol for the least significant bit, that is, we will show that a code will never code an even bit and at the same time an odd bit.

Proof. Let X, Y, P, g, a, b, c, d be strings of length n. Interpret g as if it is an element of $\mathbb{Z}_P^*$ and especially let all computations be with respect to $\mathbb{Z}_P^*$.

Then let A_0 be the statement:

$$g^a = X \wedge g^b = Y \wedge g^{a \cdot b} \text{ is even}$$

and A_1 be the statement:

$$g^c = X \wedge g^d = Y \wedge g^{c \cdot d} \text{ is odd.}$$

Then a proof of correctness with respect to the least significant bit is a proof of $\neg(A_0 \wedge A_1)$.

We can prove the unsatisfiability of A_0 and A_1 as follows.

$$\begin{aligned}
& g^{ab} \mod (P) \\
&= (g^a \mod (P))^b \mod (P) \\
&= X^b \mod (P) \\
&= (g^c \mod (P))^b \mod (P) \\
&= g^{cb} \mod (P) \\
&= g^{bc} \mod (P) \\
&= (g^b \mod (P))^c \mod (P) \\
&= Y^c \mod (P) \\
&= (g^d \mod (P))^c \mod (P) \\
&= g^{dc} \mod (P) \\
&= g^{cd} \mod (P).
\end{aligned} \tag{4.1}$$

As A_0 claims that $g^{a \cdot b}$ is even, while A_1 claims that $g^{c \cdot d}$ is odd, $A_0 \wedge A_1$ is not satisfiable. □

To do this proof in $\mathbf{VTC}^0$ we have to show that for a fixed generator g we can verify the following rules of exponentiation:

- $g^{ab} \mod (P) = (g^a \mod (P))^b \mod (P)$.
- $g^{ab} \mod (P) = g^{ba} \mod (P)$.
- We have to be able to tell whether $g^x = X$ for some fixed X and arbitrary x.
- We have to be able to tell the parity of $g^x \mod (P)$ for arbitrary x.

We will start by formalizing a version of exponentiation of strings modulo a string and work in $\mathbf{VTC}^0$ with this exponentiation added axiomatically. More precisely we want to formalize the relation $Y = (G^X)^Z \mod P$ by a formula $\mathsf{exp}_{G,P}(Y, X, Z)$. To this end let $\mathsf{exp}_{G,P}(Y, X, Z)$ be the universal closure of the conjunction of the following formulas:

1. $\mathsf{exp}_{G,P}(1, 0, Z)$
2. $\mathsf{exp}_{G,P}(1, X, 0)$
3. $\mathsf{exp}_{G,P}(Y, X + 1, 1) \wedge \mathsf{exp}_{G,P}(Y', X, 1) \rightarrow Y = Y' \cdot G \mod P$
4. $\mathsf{exp}_{G,P}(Y, X, Z + 1) \wedge \mathsf{exp}_{G,P}(Y', X, Z) \wedge \mathsf{exp}_{G,P}(Y'', X, 1) \rightarrow Y = Y' \cdot Y'' \mod P$

5. $\mathsf{exp}_{G,P}(Y, X, Z) \leftrightarrow \mathsf{exp}_{G,P}(Y, X \cdot Z, 1)$
6. $\mathsf{exp}_{G,P}(Y, X, Z) \leftrightarrow \mathsf{exp}_{G,P}(Y, 1, X \cdot Z)$
7. $\mathsf{exp}_{G,P}(Y, X + X', Z) \wedge \mathsf{exp}_{G,P}(Y', X, Z) \wedge \mathsf{exp}_{G,P}(Y'', X', Z) \rightarrow Y = Y' \cdot Y''$ mod P
8. $\mathsf{exp}_{G,P}(Y, X, Z + Z') \wedge \mathsf{exp}_{G,P}(Y', X, Z) \wedge \mathsf{exp}_{G,P}(Y'', X, Z') \rightarrow Y = Y' \cdot Y''$ mod P
9. $\mathsf{exp}_{G,P}(Y, X \cdot X', Z) \leftrightarrow \mathsf{exp}_{G,P}(Y, X, Z \cdot X')$
10. $\mathsf{exp}_{G,P}(Y, X, 1) \wedge \mathsf{exp}_{G,P}(Y, X', 1) \rightarrow (\mathsf{exp}_{G,P}(Y', X, Z) \leftrightarrow \mathsf{exp}_{G,P}(Y', X', Z))$
11. $A \neq B \rightarrow \neg(\mathsf{exp}_{G,P}(A, X, Z) \wedge \mathsf{exp}_{G,P}(B, X, Z))$.

We let

$$\mathsf{EXP}_{G,P} \equiv_{def} \forall X \leq t_1(n) \forall Z \leq t_3(n) \exists Y \leq t_3(n) \mathsf{exp}_{G,P}(Y, X, Z).$$

Multiplication and modulo can be Σ_1^B-defined in $\mathbf{VTC}^0$. We let

$$\mathsf{MOD}(Y, X, P) \equiv \exists Z. X = Z \cdot P + Y.$$

See [29] IX.3.6.3 for a formalization of multiplication and proofs of its main properties.

Proposition 4.2 ([29] Lemma IX.3.28)**.** $\mathbf{VTC}^0$ proves that multiplication of strings is commutative.

To work in $\mathbf{VTC}^0$ we will need an additional symbol for the multiplication function or define a predicate witnessing it. We choose to work in $\mathbf{VTC}^0(\cdot)$, which is $\mathbf{VTC}^0$ with an additional function symbol and additional axioms as in [29] IX.3.6.3, defining $\cdot$ to be the multiplication function.

Proposition 4.3. With multiplication $\mathsf{MOD}(Y, X, P)$ can be Σ_0^B-defined.

Proof. The following definition is Σ_0^B:

$$(Y = X - (|X| - |P|) \cdot P \wedge X - (|X| - |P|) \cdot P < P) \vee$$
$$(Y = X - (|X| - |P| + 1) \cdot P \wedge X - (|X| - |P|) \cdot P \geq P).$$

□

Proposition 4.4. $\mathbf{VTC}^0(\cdot) + \{\mathsf{EXP}_{G,P}\} \vdash \neg(A_0 \wedge A_1)$.

Proof. We will argue in $\mathbf{VTC}^0(\cdot) + \{\mathsf{EXP}_{G,P}\}$. By A_0 we have

$$\mathsf{exp}_{G,P}(K, A \cdot B, 1), \text{ for some even } K.$$

Now, by Rule 9 of $\mathsf{exp}_{G,P}$ we get that the above is equivalent to

$$\mathsf{exp}_{G,P}(K, A, B).$$

Now, from A_0 and A_1 we have

$$\mathsf{exp}_{G,P}(X, A, 1) \text{ and } \mathsf{exp}_{G,P}(X, C, 1).$$

Therefore, an application of Rule 10 yields

$$\mathsf{exp}_{G,P}(K, C, B).$$

An applications of Rule 5, followed by by commutativity of multiplication and another application of Rule 5 yields

$$\mathsf{exp}_{G,P}(K, B, C).$$

From A_0 and A_1 we get

$$\mathsf{exp}_{G,P}(Y, B, 1) \text{ and } \mathsf{exp}_{G,P}(Y, D, 1)$$

and, using Rule 10 another time,

$$\mathsf{exp}_{G,P}(K, D, C).$$

Rule 5 together with commutativity of multiplication now yields

$$\mathsf{exp}_{G,P}(K, C \cdot D, 1).$$

From A_1 we can deduce

$$\mathsf{exp}_{G,P}(K', C \cdot D, 1) \text{ for some odd } K'.$$

This, however, contradicts Rule 11. □

4.1.1 Proving $\mathsf{EXP}_{G,P}$

The proof of Proposition 4.4 can only be turned into a TC^0 -Frege proof if one adds a witness for exponentiation, for example a look-up table. The drawback is that this is an exponential sized object and the proof size will therefore explode. To circumvent this problem we will try to formalize the exponentiation of a given generator G of $\mathbb{Z}_P^*$ in $\mathbf{VTC}^0(\cdot)$ using iterated multiplication and a table only for iterated squaring of G and then use this to carry out the above argument.

To this end we let

$$\mathsf{sq}_{G,P}(S) \equiv \forall i \leq n(S[0] = 1 \wedge S[1] = G \wedge (i > 1 \rightarrow S[i+1] = S[i] \cdot S[i] \mod P))$$

and

$$\mathsf{SQ}_{G,P} \equiv \exists S \leq t(n)\mathsf{sq}_{G,P}(S),$$

where the bound $t(n)$ stems from the sequence encoding. Also

$$\begin{aligned}\mathsf{dsq}_{G,P}(S) \equiv \forall i, j \leq n(&S[0,0] = G \wedge \\ &(j > 0 \rightarrow S[0, j+1] = S[j] \cdot S[j] \mod P) \wedge \\ &(i > 0 \wedge j > 0 \rightarrow S[i+1, j] = S[i, j] \cdot S[i, j] \mod P))\end{aligned}$$

and

$$\mathsf{DSQ}_{G,P} \equiv \exists S \leq t(n)\mathsf{dsq}_{G,P}(S).$$

Thus, $\mathsf{SQ}_{G,P}$ says that the sequence $\langle 1, G^1, G^2 \mod P, \ldots, G^{2^n} \mod P\rangle$ exists and $\mathsf{DSQ}_{G,P}$ says that the matrix $\langle (G^{2^i})^{2^j} \mod P\rangle$ exists. Moreover, $\mathsf{dsq}_{G,P}$ and $\mathsf{sq}_{G,P}$ are Σ^B_0-definable in $\mathbf{VTC}^0$ from one another. Therefore, if $\mathsf{SQ}_{G,P}$ holds, so does $\mathsf{DSQ}_{G,P}$ and vice versa.

Additionally, we want to postulate for any prime P and any sequence $(S_i)_{i<n}$ of strings of length at most n the existence of a string PROD, such that

$$\mathsf{PROD} = \prod_i^n (S_i \mod P) \mod P.$$

Thus, we let

$$\begin{aligned}\mathsf{prod}(S,P) = T \equiv &\forall i \leq n (T[0] = S[0] \wedge \\ &T[i+1] = T[i] \cdot S[i+1] \mod P\end{aligned}$$

and

$$\mathsf{Prod}_{S,P} \equiv \exists T \leq t(n) \mathsf{prod}(S,P) = T.$$

Additionally we want $\mathsf{Prod}_{S,P}$ to obey certain rules of the product, such as

1. Uniqueness

$$\forall S, P, T, T' (\mathsf{prod}(S,P) = T \wedge \mathsf{prod}(S,P) = T' \rightarrow T = T')$$

2. Commutativity

$$\begin{aligned}\forall P, S, S', T, T' ((\forall i \exists j S[i] = S'[j]) \wedge (\forall i \exists j S'[i] = S[j]) \\ \wedge \mathsf{prod}(S,P) = T \wedge \mathsf{prod}(S',P) = T' \rightarrow T = T')\end{aligned}$$

3. Compatibility with Multiplication If $S = S' \frown S''$, where $\frown$ denotes concatenation, then
$$\mathsf{prod}(S,P) = \mathsf{prod}(S',P) \cdot \mathsf{prod}(S'',P).$$

$\mathsf{PROD}_{S,P}$ will be the conjunction of $\mathsf{Prod}_{S,P}$ and the above rules, PROD_P the universal closure with respect to S.

We can show in $\mathbf{V}^0$ with a the product axioms added, that a product can be evaluated step by step, that is, the following proposition holds.

Proposition 4.5. $\mathbf{VTC}^0(\cdot) + \mathsf{PROD}_P \vdash \forall i \leq |S| \, (\mathit{length}(S) - 1 = \mathit{length}(S') \wedge S'[0] = S[0] \cdot S[1] \wedge i > 1 \rightarrow S[i] = S'[i-1])) \rightarrow \mathsf{prod}(S',P) = \mathsf{prod}(S,P)$.

Proof. By induction on $\mathit{length}(S)$ and the definition of prod. □

For a string S and a string T coding a sequence of length $|S|$ we let the *restriction of* T *to* S, $T \restriction S$, be the string T' with

$$\forall i < |S| \, (S_i \rightarrow T'[i] = T[i] \wedge \neg S_i \rightarrow T'[i] = 1).$$

For a string T coding a matrix of size $|S \cdot S'|$ we let the *restriction of* T *to* S, S', $T \restriction (S, S')$, be the string T' with

$$\forall i, j < |S| \, (S_i \cdot S'_j \rightarrow T'[i,j] = T[i,j] \wedge \neg S_i \cdot S'_j \rightarrow T'[i,j] = 1).$$

We can now define what it means for a string G to be a generator of $\mathbb{Z}_P^*$. Let

$$\mathsf{GEN}(G, G', P) \equiv \mathsf{sq}_{G,P}(G') \wedge \forall S, S' < P \neg \mathsf{prod}(G' \restriction S, P) = \mathsf{prod}(G' \restriction S', P).$$

That is, $G^I \mod P$ is different for all $I < P$.

A string P represents a prime iff

$$\mathsf{PRIME}(P) \equiv \forall S, S' < P(S \cdot S' \neq P)$$

holds. We now claim the following.

Proposition 4.6. $\mathbf{VTC}^0(\cdot) + \mathsf{PROD}_P + \mathsf{SQ}_{G,P} \vdash (\mathsf{PRIME}(P) \wedge \mathsf{GEN}(G, G', P) \rightarrow \mathsf{EXP}_{G,P})$.

Proof. We have to give a definition of $\mathsf{EXP}_{G,P}$ within our setting and prove its properties. To this end we let

$$\mathsf{exp}^*{}_{G,P}(X, Y, Z) \equiv \mathsf{dsq}_{G,P}(G') \wedge X = \mathsf{prod}(G' \restriction (Y, Z), P).$$

We can justify this definition as in $\mathbb{N}_2$ the following holds. For any string X let $bin(X) := \sum_i 2^i X_i$ the number X represents in binary. Then

$$\begin{aligned}
(G^X)^Y &= (bin(G)^{bin(X)})^{bin(Y)} \\
&= (bin(G)^{\sum_i 2^i X_i})^{\sum_j 2^j Y_j} \\
&= bin(G)^{(\sum_i 2^i X_i) \cdot (\sum_j 2^j Y_j)} \\
&= bin(G)^{\sum_i \sum_j (2^i \cdot 2^j X_i \cdot Y_j)} \\
&= bin(G)^{\sum_i \sum_j (2^{i+j} X_i \cdot Y_j)} \\
&= \prod_{i,j} bin(G)^{2^{i+j} X_i Y_j}.
\end{aligned}$$

We will show now that $\mathsf{exp}^*{}_{G,P}$ fulfills all items of the definition of $\mathsf{exp}_{G,P}$ (4.1):

Let G' be such that $\mathsf{dsq}_{G,P}(G')$ holds.

1. $\mathsf{prod}(G' \restriction (0, Z), P) = \mathsf{prod}(1, P) = 1$.
2. $\mathsf{prod}(G' \restriction (Y, 0), P) = \mathsf{prod}(1, P) = 1$.
3. follows from Compatibility with Multiplication.
4. We have to show that

   $$\mathsf{prod}(G' \restriction (Y, Z+1), P) = \mathsf{prod}(G' \restriction (Y, Z), P) \cdot \mathsf{prod}(G' \restriction (Y, 1), P).$$

   We will prove this by induction on Z. If $Z = 0$ the statement follows from item 2 above. Now, for $Z > 0$ we take a look at the columns c_i of the matrix $G' \restriction (Y, Z)$ where $Y_i = 1$. Inductively on the index of the first 0 in Z, we can use Proposition 4.5.
5. follows straightforward from the definition.
6. follows straightforward from the definition.

7. We have to show that

$$\mathsf{prod}(G' \restriction (Y, Z), P) \cdot \mathsf{prod}(G' \restriction (Y', Z), P) = \mathsf{prod}(G' \restriction (Y + Y', Z), P).$$

   We proceed by induction of Y'. If $Y' = 0$ the statement follows from item 1. For the induction step we can apply item 3. It remains to show that

$$\mathsf{prod}(G' \restriction (Y + (Y' + 1), Z), P) = \mathsf{prod}(G' \restriction (Y + Y', Z), P) \cdot G \mod P.$$

   The statement now follows from Commutativity and Compatibility with Multiplication.

8. analogously as in the previous item using items 2 and 4 instead of 1 and 3.

9. analogously to 5.

10. by induction on Z and 4.

11. by uniqueness of the product.

□

The overall goal is now to further reduce the axioms to eventually reach a minimal theory for a) modulo exponentiation or b) for a correctness proof of DH. The former statement is interesting for finding a proper theory for TC^0, as we are well within this complexity class, yet it seems hard to formalize what we did above purely in $\mathbf{VTC}^0$. The latter is interesting from a more applied view, as was explained in the beginning of this chapter. Also, if we could devise a weak theory that proves the correctness of DH, we might be able to develop a uniform proof of the main result of [18]. Unfortunately I was so far unable to produce anything noteworthy to further any of these approaches.

5. Conclusion

We will recapture the major results of this thesis now and later give an outlook towards ways to improve them. Basically, the results can be perceived in two different ways. On the one hand they comprise results about bounded arithmetic theories and their strength. On the other hand, they consist of results about efficient provability in propositional proof systems and therefore also about separations of such systems. We will treat them separately.

Concerning bounded arithmetic, the main result is Theorem 3.6 (Theorem B.13 in Appendix B), which states that a formalized version of Nepomnjascij's Theorem holds for polylogarithmic cuts of models of $\mathbf{V}^0$. That result is interesting, mainly because it allows for plentiful applications concerning the definability of small sets in models in $\mathbf{V}^0$. In some sense the algorithmic view might be perceived to be more natural, as it seems to be better adaptable to our way of thinking. For example, it is straightforward to give an algorithm that computes the size of a given set, yet the definition of this is quite tedious in weak bounded arithmetic. So, by our result, straightforwardly $\mathbf{V}^0$ can tell the size of polylogarithmic sized sets (and thus its polylogarithmic cut is a model of $\mathbf{VTC}^0$), while it is non-trivial to derive that result directly (see Paris and Wilkie [74, 75]).

Concerning proof complexity our results are more manifold. Theorem 2.1 (Theorem 1 in Appendix A) gives a separation result between Resolution and TC^0-Frege, that does not depend on any semantical properties of the tautologies, i.e. it is not important what they say, only that they have some simple structural property. In a weaker sense this is also true for Theorem 3.9 (Theorem B.17 in Appendix B) and separation between Resolution and bounded depth Frege. The proof of Theorem 2.1 also shows that sometimes it is possible to apply methods in proof systems that exceed their associated computational strength (proper reasoning about eigenvalues is presumably only possible from NC^2 onwards). This is mainly due to the non determinism of proofs. It suffices to know that there is an object that satisfies a needed condition, e.g. being an eigenvalue and therefore having certain properties, to be able to use it in a proof. This certainly does not yield uniform proofs, but the non-uniformity only stems from certain substitutions.

Apart from the aforementioned separation results we also obtain a new proof of the sub exponential simulation result due to Filmus, Pitassi and Santhanam [39] between Frege and bounded depth Frege. Moreover, analyzing the theory $\mathbf{VNC}$ should yield a strengthening of Frege that is still sub exponentially simulated by bounded depth Frege systems. This is in the light of Section 3.2 in Chapter 3 of this thesis.

Outlook

There are a few natural ideas of how to extend the results from this thesis. Concerning the results about polylogarithmic cuts, it would be interesting to determine what theory is necessary as a base theory, to conclude that the cut is a model of $\mathbf{VP}$. The latter theory is rather strong and corresponds to extended Frege systems. That is, to Frege systems that make use of extension axioms.

Supposedly, **VNC** is still weaker, so it might be a good starting point as a base theory. Unfortunately I do not see yet, how or if the additional strength gives rise to a stronger version of Nepomnjascij's Theorem, i.e. one, where stronger means of computation already yield definability, or whether they grant any additional strength to algorithms that can be defined using the proved formalization. Another interesting direction would be the opposite: What happens if we weaken the base theory? Can we still get some sensible results? I do not have much intuition about this, but this could prove to be very interesting in the question about the non-automatizability of Resolution, as was already discussed in Chapter 4.

A straightforward way of strengthening Theorem 2.1 would be to try to argue in a weaker theory than $\mathbf{VTC}^0$. As we already use approximations for the rational numbers and thus do all our counting arguments with approximated values, it is rather natural to assume that we do not need a proper version of counting, but could do with some sort of approximated counting, as was introduced by Jeřábek [55]. Unfortunately computing with approximations of rational numbers is only part of the story. We need to explicitly be able to add modulo 2, which cannot be done using approximations. Thus we probably can weaken the theory to a theory of approximated counting, but we would have to explicitly add an axiom for counting modulo 2. This should still be interesting, though. Another way, proposed by Iddo Tzameret, would be to formalize a weaker result than [37], namely [38]. This should be doable using only approximated counting (and thus would correspond to bounded depth Frege), but the clause-to-variable ratio we could prove the result for would not suffice to conclude the separation from Resolution.

Concerning the computability of certain functions, especially modulo exponentiation and modulo iterated multiplication, it would be very interesting to devise a weak natural theory in which that is possible and which also proves certain properties of that function. Such a theory might be a good candidate for a natural formalization of the strength of the circuit class TC^0. I would also find a result which only considers calculations with respect to a fixed generator most interesting, as this might allow for the use of cryptographic assumptions to obtain other results. It would certainly be interesting to produce a proof that shows that $\mathbf{VTC}^0$ is not strong enough to do so.

Bibliography

[1] M. Ajtai. The Complexity of the Pigeonhole Principle. *Proceedings of FOCS*, 1988, pp.346–355.

[2] M. Alekhnovich. Lower Bounds for k-DNF Resolution on Random 3-CNFs. *Proceedings of STOC*, 2005, pp.251—256.

[3] M. Alekhnovich and A. A. Razborov. Lower Bounds for Polynomial Calculus: Non-Binomial Case. *Proceedings of FOCS*, 2001, pp.190—199.

[4] M. Alekhnovich and A. A. Razborov. Resolution is not Automatizable Unless W[P] is Tractable. *SIAM Journal of Computing*, Vol. **38**(4), 2008, pp.1347—1363.

[5] S. Arora and B. Barak. Computational Complexity. *Cambridge University Press*, 2009.

[6] A. Atserias and M. L. Bonet. On the Automatizability of Resolution and Related Propositional Proof Systems. *Information and Computation*, Vol. **189**(2), 2004, pp. 182–201.

[7] A. Atserias, M. L. Bonet and J. Esteban. Lower Bounds for the Weak Pigeonhole Principle and Random Formulas Beyond Resolution. *Information and Computation*, Vol. **176**, 2002.

[8] J. Barwise. Handbook of Mathematical Logic. *Studies in Logic and the Foundations of Mathematics*, Vol. **90**, Elsevier, 1977.

[9] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák, and A. Woods. Exponential lower bounds for the pigeonhole principle. *Proceedings of the 24th STOC*, 1992, pp.200–220.

[10] P. Beame, R. Karp, T. Pitassi and M. Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, Vol. **31**(4), 2002, pp.1048–1075.

[11] J. H. Bennett. On Spectra. *PhD thesis*, Princeton University, 1962.

[12] E. Ben-Sasson. Expansion in Proof Complexity. PhD Thesis, Hebrew University, Jerusalem, 2001.

[13] E. Ben-Sasson and Y. Bilu. A Gap in Average Proof Complexity. *ECCC*, 2002.

[14] E. Ben-Sasson and R. Impagliazzo. Random CNF's are hard for the Polynomial Calculus. *Proceedings of FOCS*, 1999.

[15] E. Ben-Sasson and A. Wigderson. Short proofs are narrow—resolution made simple. *Proceedings of STOC*, 1999, pp.517–526.

[16] A. Blake. Canonical Expressions in Boolean Algebra. PhD Thesis, University of Chicago, 1937.

[17] M.L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, and T. Pitassi. Non-Automatizability of Bounded-Depth Frege Proofs. *Computational Complexity*, Vol. **13**, 2004, pp.47–68.

[18] M.L. Bonet, T. Pitassi, and R. Raz. On interpolation and automatization for Frege systems. *SIAM Journal of Computing*, Vol. **29 (6)**, 2000, pp. 1939–1967.

[19] J. Buresh-Oppenheim, P. Beame, T. Pitassi, R. Raz and A. Sabharwal. Bounded-depth Frege Lower Bounds for Weaker Pigeonhole Principles. *Proceedings 43rd Annual Symposium on Foundations of Computer Science*, 2002.

[20] S. R. Buss. Bounded Arithmetic. *Studies in Proof Theory*, Vol. **3**, Bibliopolis, 1986.

[21] S. R. Buss. Polynomial Size Proofs of the Propositional Pigeonhole Principle. *Journal of Symbolic Logic*, Vol. **52**, 1987, pp. 916–927.

[22] S. R. Buss (ed.) Handbook of Proof Theory *Studies in Logic and the Foundations of Mathematics*, Vol. **137**, Elsevier, Amsterdam, 1998.

[23] S. R. Buss and P. Clote. Cutting Planes, Connectivity, and Threshold Logic. *Archives of Mathematical Logic*, Vol. **35**(1), 1996, pp.33-–62.

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

[25] V. Chvátal and E. Szemerédi. Many Hard Examples for Resolution. *Journal of the Association for Computing Machinery*, Vol. **35**(4), 1988, pp. 759–768.

[26] S. Cook. The complexity of theorem proving procedures. In *Proceedings of 3rd ACM SToC*, 1971, pp.151-–158.

[27] S. Cook. Theories for Complexity Classes and Their Propositional Translations. *Complexity of Computations and Proofs*, Quaderni di Matematica, 2005, 175–227.

[28] S. Cook, K. Ghasemloo and P. Nguyen. Subexponential Size Bounded Depth Frege Proofs and Subsystems of TV^0. Manuscript, 2012.

[29] S. Cook and P. Nguyen. Logical Foundations of Proof Complexity. *Cambridge University Press*, 2010.

[30] S. Cook and R. Reckhow. The Relative Efficiency of Propositional Proof Systems. *Journal of Symbolic Logic*, Vol. **44**(1), 1979, pp.36–50.

[31] M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem Proving. Communications of the ACM, Vol. **5**(7), 1962.

[32] M. Davis and H. Putnam. A computing Procedure for Quantification Theory. *Journal of the ACM*, Vol. **7**(3), 1960, pp.201—215.

[33] R. Dedekind. *Was sind und was sollen die Zahlen?* Vieweg, Braunschweig, 1888.

[34] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, Vol. **IT-22**, 1976, pp.644–654.

[35] H.-D. Ebbinghaus, J. Flum and W. Thomas. Mathematical Logic. Undergraduate Texts in Mathematics, Springer, 1984.

[36] U. Feige. Refuting Smoothed 3CNF Formulas. *Proceedings of FOCS*, 2007, pp.407—417.

[37] U. Feige, J. H. Kim and E. Ofek. Witnesses for non-satisfiability of dense random 3CNF formulas. *Proceedings of FOCS*, 2006.

[38] U. Feige and E. Ofek. Easily Refutable Subformulas of Large Random 3CNF Formulas. *Theory of Computing*, Vol. **3**(1), 2007, pp.25—43.

[39] Y. Filmus, T. Pitassi, and R. Santhanam. Exponential Lower Bounds for AC-Frege Imply Superpolynomial Frege Lower Bounds. *Proceedings of ICALP*, Vol. **1**, 2011, pp.618–629.

[40] G. Frege. Die Grundlagen der Arithmetik. Köbner, Breslau, 1884.

[41] J. Friedman, A. Goerdt, and M. Krivelevich. Recognizing More Unsatisfiable Random k-SAT Instances Efficiently. *SIAM Journal for Computing*, Vol. **35**(2), 2005, pp.408—430.

[42] N. Galesi and M. Lauria. Optimality of size-degree trade-offs for polynomial calculus. *ACM Transactions of Computational Logic*, Vol. **12**(1), 2011.

[43] G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, Vol. **39**(2), 1934, pp.176-–210.

[44] G. Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, Vol. **39**(3), 1935, pp.405-–431.

[45] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, Vol. **38**, 1931, pp.173-–198.

[46] A. Goerdt and M. Krivelevich. Efficient Recognition of Random Unsatisfiable k-SAT Instances by Spectral Methods. *Proceedings STACS*, 2001, pp.294–304.

[47] A. Goerdt and A. Lanka. Recognizing More Random Unsatisfiable 3-SAT Instances Efficiently. *Electronic Notes in Discrete Mathematics*, Vol. **16**, 2003, pp.21—46.

[48] H. Grassmann. *Lehrbuch der Arithmetik*. Enslin, Berlin, 1861.

[49] P. Hajek and P. Pudlák. Metamathematics of First-order Arithmetic. *Perspectives in Mathematical Logic*, Springer-Verlag, Berlin, 1993.

[50] A. Haken. The Intractability of Resolution. *Theoretical Computer Science*, Vol. **39**, 1985, pp.297–308.

[51] W. Hesse, E. Allender and D. Barrington. Uniform Constant-Depth Threshold Circuits for Division and Iterated Multiplication. *Journal of Computer and System Sciences*, Vol. **65**, 2002, pp.695–716.

[52] D. Hilbert. Die Grundlegung der elementaren Zahlenlehre. *Mathematische Annalen*, Vol. **104**, 1931, pp.485–94.

[53] R. A. Horn and C. R. Johnson. Matrix Analysis. *Cambridge University Press*, 1985.

[54] E. Jeřábek. Dual Weak Pigeonhole Principle, Boolean Complexity, and Derandomization. *Annals of Pure and Applied Logic*, Vol. **129**, 2004, pp.1–37.

[55] E. Jeřábek. Approximate counting by hashing in bounded arithmetic. *Journal of Symbolic Logic*, Vol. **74**(3), 2009, pp. 829-–860.

[56] J. Krajíček. Lower Bounds to the Size of Constant-Depth Propositional Proofs. *The Journal of Symbolic Logic*, Vol. **59**(1), 1994, pp.73–86.

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

[58] J. Krajíček. Interpolation theorems, lower bounds for proof systems, and indepen- dence results for bounded arithmetic. *Journal of Symbolic Logic*, Vol. **62**(2), 1997, pp.457–486.

[59] J. Krajíček. On the Weak Pigeonhole Principle. *Fundamenta Mathematicae*, Vol. **170**, 2001, pp.123–140.

[60] J. Krajíček. Forcing with Random Variables in Bounded Arithmetic and Proof Complexity. *London Mathematical Society Lecture Note Series*, Vol.**382**, Cambridge University Press, 2011.

[61] J. Krajíček and P. Pudlák. Some Consequences of Cryptographical Conjectures for S_2^1 and EF. *Information and Computation*, Vol. **140 (1)**, 1998, pp.82–94.

[62] J. Krajíček, P. Pudlák and A. Woods. Exponential lower bound to the size of bounded depth Frege proofs of the Pigeon Hole Principle. *Random Structures and Algorithms*, Vol. **7**(1), 1995, pp.15–39.

[63] L. Levin. Universal sequential search problems. *PINFTRANS: Problems of Information Transmission (translated from Problemy Peredachi Informatsii (Russian))*, Vol. **9**, 1973.

[64] P. Lindström. On Existensions of Elementary Logic. Theoria, Vol. **35**, 1969, pp.1–11.

[65] A. Maciel and T. Pitassi. On $ACC^0[p^k]$ Frege Proofs. *Proceedings of STOC*, 1997, pp.720—729.

[66] S. Müller. Polylogarithmic Cuts in Models of $\mathbf{V}^0$. submitted, 2012.

[67] S. Müller and I. Tzameret. Short Propositional Refutations for Dense Random 3CNF Formulas. *Proceedings of LICS*, 2012.

[68] V.A. Nepomnjascij. Rudimentary Predicates and Turing Calculations. *Doklady AN SSSR*, Vol. **195**, 1970.

[69] P. Nguyen. Proving Infinitude of Prime Numbers Using Binomial Coefficients. *Proceedings of CSL*, 2008.

[70] P. Nguyen and S. Cook. Theories for $\mathsf{TC}^0$and other Small Complexity Classes. *Logical Methods in Computer Science*, Vol. **2**(1), 2006.

[71] R. Parikh. Existence and Feasibility in Arithmetic. *Journal of Symbolic Logic*, Vol. **36**, 1971, pp.494–508.

[72] J. Paris and C. Dimitracopoulos. A Note on the Undefinability of Cuts. *Journal of Symbolic Logic*, Vol. **48**(3), 1983, pp.564–569.

[73] J. Paris and A. Wilkie. Δ_0 Sets and Induction. *Open Days in Model Theory and Set Theory*, 1981, pp.237—248.

[74] J. Paris and A. Wilkie. Counting Problems in Bounded Arithmetic. *Methods in Mathematical Logic*, LNM **1130**, 1985, pp.317–340.

[75] J. Paris and A. Wilkie. Counting Δ_0 Sets. *Fundamenta Mathematica*, Vol. **127**, 1987, pp.67–76.

[76] J. Paris, A. Wilkie, and A. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *Journal of Symbolic Logic*, Vol.**53**, 1988, pp.1235-–1244.

[77] G. Peano. *Arithmetices principia, nova methodo exposita.* Bocca, Torino, 1889.

[78] T. Pitassi, P. Beame, R. Impagliazzo. Exponential Lower Bounds for the Pigeonhole Principle. *Computational Complexity*, Vol. **3**, 1993, pp.97–140.

[79] P. Pudlák. A Definition of Exponentiation by a Bounded Arithmeic Formula. *Commentationes Mathematicae Universitas Carolinae*, Vol. **24**(4), 1983, 667–671.

[80] P. Pudlák. Logical Foundations of Mathematics and Computational Complexity – A Gentle Introduction. *to appear.*

[81] R. Raz. Resolution Lower Bounds for the Weak Pigeonhole Principle. *Journal of the ACM*, Vol. **51**(2), 2004, pp.115–138.

[82] A. Razborov. Improved Lower Bounds for the Weak Pigeonhole Principle. *ECCC*, 2001.

[83] R. Reckhow. On the Lengths of Proofs in the Propositional Calculus. PhD Thesis, University of Toronto, Toronto, 1976.

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

[85] J. Rosser. Extensions of some Theorems of Gödel and Church. *Journal of Symbolic Logic*, Vol. **1**, 1936, pp.87–91.

[86] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, Vol. **4**, 1970, pp.177—192.

[87] N. Segerlind, S. Buss and R. Impagliazzo. A Switching Lemma for Small Restrictions and Lower Bounds for k-DNF Resolution. *SIAM Journal of Computing*, Vol. **33**(5), 2004, pp.1171-–1200.

[88] R. Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. *Proceedings of STOC*, 1987, pp.77–82.

[89] M. Soltys and S. Cook. The Proof Complexity of Linear Algebra. *Annals of Pure and Applied Logic*, Vol. **130**, 2004, pp.277–323.

[90] H. Vollmer. Introduction to Circuit Complexity. *Texts in Theoretical Computer Science*, Springer, 1999.

[91] R. Zach. Hilbert's Program. *The Stanford Encyclopedia of Philosophy (Spring 2009 Edition)*, URL=¡http://plato.stanford.edu/archives/spr2009/entries/hilbert-program/¿, 2009.

[92] D. Zambella. Notes on Polynomially Bounded Arithmetic. *Journal of Symbolic Logic*, Vol. **61**(3), 1996, pp.942–966.

A. Short Refutations for Random 3CNF

This paper deals with the average complexity of propositional proofs. Our aim is to show that standard propositional proof systems, within the hierarchy of Frege proofs, admit short random 3CNF refutations and can outperform resolution for almost all unsatisfiable 3CNF formulas, for a sufficiently large clause-to-variable ratio. Specifically, we show that most 3CNF formulas with n variables and at least $cn^{1.4}$ clauses, for a sufficiently large constant c, have polynomial-size in n propositional refutations whose proof-lines are constant depth circuits with threshold gates (namely, TC^0-Frege proofs). This is in contrast to resolution (that can be viewed as depth-1 Frege) for which it is known that most 3CNF formulas with at most $n^{1.5-\epsilon}$ clauses (for $0 < \epsilon < \frac{1}{2}$) do not admit sub-exponential refutations [25, 15].

The main technical contribution of this paper is a propositional characterization of the random 3CNF unsatisfiability witnesses given by Feige at al. [37]. In particular we show how to carry out certain spectral arguments inside weak propositional proof systems at least as strong as TC^0-Frege. The latter should hopefully be useful in further propositional formalizations of spectral arguments. This also places a stream of recent works on efficient refutation algorithms using spectral arguments—beginning in the work of Goerdt and Krivelevich [46] and culminating in Feige et al. [37]—within the framework of propositional proof complexity. Loosely speaking, we show that all these refutation algorithms and witnesses, considered from the perspective of propositional proof complexity, are not stronger than TC^0-Frege.

A.0.2 Background in proof complexity

Propositional proof complexity is the systematic study of the efficiency of proof systems establishing propositional tautologies (or dually, refuting unsatisfiable formulas). *Abstractly* one can view a propositional proof system as a deterministic polynomial-time algorithm A that receives a string π ("the proof") and a propositional formula Φ such that there exists a π with $A(\pi, \Phi) = 1$ iff Φ is a tautology. Such an A is called an *abstract proof system* or a *Cook-Reckhow proof system* due to [30]. Nevertheless, most research in proof complexity is dedicated to more concrete or structured models, in which proofs are sequences of lines, and each line is derived from previous lines by "local" and sound rules.

Perhaps the most studied family of propositional proof systems are those coming from propositional logic, under the name Frege systems, and their fragments (and extensions). In this setting, proofs are written as sequences of Boolean formulas (proof-lines) where each line is either an axiom or was derived from previous lines by means of simple sound derivation rules. The *complexity* of a proof is just the number of symbols it contains, that is, the total size of formulas in it. And different proof systems are compared via the concept of *polynomial simulation*: a proof system P polynomially-simulates another proof system Q if there is a polynomial-time computable function f that maps Q-proofs to P-proofs

of the same tautologies. The definition of Frege systems is sufficiently robust, in the sense that different formalizations can polynomially-simulate each other [83].

It is common to consider fragments (or extensions) of Frege proof systems induced by restricting the proof-lines to contain presumably weaker (or stronger) circuit classes than Boolean formulas. The stratification of Frege proof systems is thus analogous to that of Boolean circuit classes: Frege proofs consist of Boolean formulas (i.e., $\mathbf{NC}^1$) as proof-lines, TC^0-Frege (also known as Threshold Logic) consists of TC^0 proof-lines, Bounded Depth Frege has $\mathbf{AC}^0$ proof-lines, depth-d Frege has circuits of depth-d proof-lines, etc. In this framework, the resolution system can be viewed as *depth*-1 *Frege*. Similarly, one usually considers extensions of Frege system such as $\mathbf{NC}^i$-Frege, for $i > 1$, and $\mathbf{P/poly}$-Frege (which is in fact the Extended Frege system as shown in [54]). Restrictions (and extensions) of Frege proof systems form a hierarchy with respect to polynomial-simulations, though it is open whether the hierarchy is proper.

It thus constitutes one of the main goals of proof complexity to understand the above hierarchy of Frege systems, and to separate different propositional proof systems, that is, to show that one proof system does not polynomially simulate another proof system. These questions also relate in certain sense to the hierarchy of Boolean circuits (from $\mathbf{AC}^0$, through, $\mathbf{AC}^0[\mathrm{p}]$, TC^0, $\mathbf{NC}^1$, and so forth; see [27]). Many separations between propositional proof systems (not just in the Frege hierarchy) are known. For this kind of results it is enough to demonstrate a single family of tautologies requiring super-polynomial proof-size in one system, while having polynomial-size proofs in the other system. In the case of Frege proofs there are already known separations between certain fragments of it (e.g., separation of depth-d Frege from depth $d+1$ Frege was shown by Krajíček [56]). It is also known that TC^0-Frege is strictly stronger than resolution and than bounded depth Frege proof system, since, e.g., TC^0-Frege admits polynomial-size proofs of the propositional pigeonhole principle, while resolution and bounded depth Frege do not (see [50] for the resolution lower bound, [1] for the bounded depth Frege lower bound and [29] for the corresponding TC^0-Frege upper bound).

Average-case proof complexity—the random 3CNF model. Much like in algorithmic research, it is important to know the average-case complexity of propositional proof systems, and not just their worst-case behavior. To this end one usually considers the model of random 3CNF formulas, where m clauses with three literals each, out of all possible $2^3 \cdot \binom{n}{3}$ clauses with n variables, are chosen independently, with repetitions. When m is greater than cn for some sufficiently large c (say, $c = 5$), it is known that with high probability a random 3CNF is unsatisfiable. (As m gets larger the task of refuting the 3CNF becomes easier since we have more constraints to use.) In average-case analysis of proofs we investigate whether such unsatisfiable random 3CNFs also have short (polynomial-size) refutations in a given proof system. The importance of average-case analysis of proof systems is that it gives us a better understanding of the complexity of a system than merely the worst-case analysis. For example, if we separate two proof systems in the average case—i.e., show that for almost all 3CNF one proof system admits polynomial-size refutations, while the other system does not—we establish a stronger separation.

Until now only weak proof systems like resolution and Res(k)

(for $k \leq \sqrt{\log n / \log\log n}$; the latter system introduced in [59] is an extension of resolution that operates with kDNF formulas) and polynomial calculus (and an extension of it) were analyzed in the random 3CNF model; for these systems exponential lower bounds are known for random 3CNFs (with varying number of clauses) [25, 10, 15, 7, 87, 2, 14, 3, 42]. For random 3CNFs with n variables and $n^{1.5-\epsilon}$ ($0 < \epsilon < \frac{1}{2}$) clauses it is known that there are no sub-exponential size resolution refutations [15]. For many proof systems, like cutting planes (CP) and bounded depth Frege ($\mathbf{AC}^0$-Frege), it is a major open problem to prove random 3CNF lower bounds (even for number of clauses near the threshold of unsatisfiability, e.g., random 3CNFs with n variables and $5n$ clauses). The results mentioned above only concerned lower bounds. On the other hand, to the best of our knowledge, the only known non-trivial polynomial-size *upper bound* on random kCNFs refutations in any non-abstract propositional proof system is for resolution. This is a result of Beame et al. [10], and it applies for fairly large number of clauses (specifically, $\Omega(n^{k-1}/\log n)$).

Efficient refutation algorithms. A different kind of results on refuting random kCNFs were investigated in Goerdt and Krivelevich [46] and subsequent works by Goerdt and Lanka [47], Friedman, Goerdt and Krivelevich [41], Feige and Ofek [38] and Feige [36]. Here, one studies efficient refutation *algorithms* for kCNFs. Specifically, an *efficient refutation algorithm* receives a kCNF (above the unsatisfiability threshold) and outputs either "unsatisfiable" or "don't know"; if the algorithm answers "unsatisfiable" then the kCNF is required to be indeed unsatisfiable; also, the algorithm should output "unsatisfiable" with high probability (which by definition, is also the correct answer). Such refutation algorithms can be viewed as *abstract* proof systems (according to the definition in Subsection A.0.2) having short proofs on the average-case: $A(\Phi)$ is a deterministic polytime machine whose input is only kCNFs (we can think of the proposed proof π input as being always the empty string). On input Φ the machine A runs the refutation algorithm and answers 1 iff the refutation algorithm answers "unsatisfiable"; otherwise, A can decide, e.g. by brute-force search, whether Φ is unsatisfiable or not. (In a similar manner, if the original efficient refutation algorithm is *non-deterministic* then we also get an abstract proof system for kCNFs; now the proof π that A receives is the description of an accepting run of the refutation algorithm.)

Goerdt and Krivelevich [46] initiated the use of *spectral methods* to devise efficient algorithms for refuting kCNFs. The idea is that a kCNF with n variables can be associated with a graph on n vertices (or directly with a certain matrix). It is possible to show that certain properties of the associated graph witness the unsatisfiability of the original kCNF. One then uses a spectral method to give evidence for the desired graph property, and hence to witness the unsatisfiability of the original kCNF. Now, if we consider a random kCNF then the associated graph essentially becomes random too, and so one may show that the appropriate property witnessing the unsatisfiability of the kCNF occurs with high probability in the graph. The best (with respect to number of clauses) refutation algorithms devised in this way work for 3CNFs with at least $\Omega(n^{1.5})$ clauses [38].

Continuing this line of research, Feige, Kim and Ofek [37] considered efficient *non-deterministic* refutation algorithms (in other words, efficient *witnesses* for

unsatisfiability of 3CNFs). They established the currently best (with respect to the number of clauses) efficient, alas non-deterministic, refutation procedure: they showed that with probability converging to 1 a random 3CNF with n variables and at least $cn^{1.4}$ clauses has a polynomial-size witness, for sufficiently big constant c.

The result in the current paper shows that all the above refutation algorithms, viewed as abstract proof systems, *are not stronger (on average) than* TC^0*-Frege*. The short TC^0-Frege refutations will be based on the witnesses from [37], and so the refutations hold for the same clause-to-variable ratio as in that paper.

A.0.3 Our result

The main result of this paper is a polynomial-size upper bound on random 3CNF formulas refutations in a proof system operating with constant-depth threshold circuits (known as Threshold Logic or TC^0-Frege; see Definition A.14). Since Frege and Extended Frege proof systems polynomially simulate TC^0-Frege proofs, the upper bound holds for these proof systems as well. (The actual formulation of TC^0-Frege is not important since different formulations, given in [23, 65, 18, 70, 29], polynomially simulate each other.)

Theorem 1. With probability $1-o(1)$ a random 3CNF formula with n variables and $cn^{1.4}$ clauses (for a sufficiently large constant c) has polynomial-size TC^0-Frege refutations.

Beame, Karp, Pitassi, and Saks [10] and Ben-Sasson and Wigderson [15] showed that with probability $1-o(1)$ resolution does not admit sub-exponential refutations for random 3CNF formulas when the number of clauses is at most $n^{1.5-\epsilon}$, for any constant $0<\epsilon<1/2$.[1] Therefore, Theorem 1 shows that TC^0-Frege has an exponential speed-up over resolution for random 3CNFs with at least $cn^{1.4}$ clauses (when the number of clauses does not exceed $n^{1.5-\epsilon}$, for $0<\epsilon<1/2$).

We now explain the potential significance of our work and its motivations. It is well known that most contemporary SAT-solvers are based on the resolution proof system. Formally, this means that these SAT-solvers use a backtracking algorithm that branch on a single variable and construct in effect a resolution refutation (in case the CNF instance considered is unsatisfiable). (The original backtracking algorithm DPLL constructs a *tree-like* resolution refutation [32, 31].) It was known since [25] that resolution is weak in the average case. Our work gives further impetus to the quest to build SAT-solvers based on stronger proof systems than resolution. Although there is little hope to devise polynomial-time algorithms for constructing minimal TC^0-Frege proofs or even resolution refutations (this stems from the conditional non-automatizability results for TC^0-Frege and resolution, proved in [18] and [4], respectively), practical experience shows that current resolution based SAT-solvers are quite powerful. Therefore,

[1] Beame *et al.* [10] showed such a lower bound for $n^{5/4-\epsilon}$ number of clauses (for any constant $0<\epsilon<1/4$). Ben-Sasson and Wigderson [15] introduced the size-width tradeoff that enabled them to prove an exponential lower bound for random 3CNF formulas with $n^{1.5-\epsilon}$ number of clauses (for any constant $0<\epsilon<1/2$), but the actual proof for this specific clause-number appears in [12].

our random 3CNF upper bound theoretically justifies an attempt to extend SAT-solvers beyond resolution.

Our result also advances the understanding of the relative strength of propositional proof systems: proving non-trivial upper bounds clearly rules out corresponding lower bounds attempts. We conjecture that random 3CNF upper bounds similar to Theorem 1 could be achieved even for systems weaker than TC^0-Frege on the expense of at most a quasipolynomial increase in the size of proofs. This might help in understanding the limits of known techniques used to prove random 3CNFs lower bounds on resolution and Res(k) refutations.

The main result also contributes to our understanding of probabilistic refutation algorithms: we give an explicit logical characterization of the Feige et al. [37] witnesses. This places a stream of results on probabilistic refutation algorithms using spectral methods, starting from Goerdt and Krivelevich [46], in the propositional proof complexity setting. This is a non-trivial job, especially because of the need to propositionally simulate spectral arguments. Moreover, our formalization of the spectral argument and its short propositional proofs might help in formalizing different arguments based on spectral techniques (e.g., reasoning about expander graphs).

A.0.4 Relations to previous works

The proof complexity of random 3CNF formulas have already been discussed above: for weak proof systems like resolution and Res(k) there are known exponential lower bounds with varying number of clauses [15]; with regards to upper bounds, there are known polynomial size resolution refutations on random 3CNF formulas with $\Omega(n^2/\log n)$ number of clauses [10]. Here we discuss several known upper and lower bounds on refutations of *different* distributions than the random 3CNF model. (This is not an exhaustive list of all distributions studied.)

Ben-Sasson and Bilu [13] have studied the complexity of refuting random 4-Exactly-Half SAT formulas. This distribution is defined by choosing at random m clauses out of all possible clauses with 4 literals over n variables. A set of clauses is *4-exactly-half satisfiable* iff there is an assignment that satisfies exactly two literals in each clause. It is possible to show that when $m = cn$, for sufficiently large constant c, a random 4-Exactly-Half SAT formulas with m clauses and n variables is unsatisfiable with high probability. Ben-Sasson and Bilu [13] showed that almost all 4-Exactly-Half SAT formulas with $m = n \cdot \log n$ clauses and n variables do not have sub-exponential resolution refutations. On the other hand, [13] provided a polynomial-time refutation algorithm for 4-Exactly-Half SAT formulas.

Other possible distribution on unsatisfiable formulas for which one can obtain a separation in the average case between two (non-abstract) proof systems is 3-LIN formulas over the two element field $\mathbb{F}_2$, or equivalently 3XOR formulas. A 3-LIN formula is a collection of linear equations over $\mathbb{F}_2$, where each equation has precisely three variables. When the number of randomly chosen linear equations with 3 variables is large enough, one obtains that with high probability the collection is unsatisfiable (over $\mathbb{F}_2$). It is possible to show that the polynomial calculus proof system, as well as TC^0-Frege, can efficiently refute such random instances with high probability, by simulating Gaussian elimination.

A Different type of distribution over unsatisfiable CNF formulas can possibly be constructed from the formulas (termed *proof complexity generators*) in Krajíček [60]. We refer the reader to [60] for more details on this.

A.0.5 The structure of the argument

Here we outline informally (and in some places in a simplified manner) the structure of the proof of the main theorem. We need to construct certain TC^0-Frege proofs. Constructing such propositional proofs directly is technically cumbersome, and so we opt to construct it indirectly by using a first-order (two-sorted) characterization of (short proofs in) TC^0-Frege: we use the theory $\mathbf{VTC}^0$ introduced in [70] (we follow tightly [29]). When restricted to proving only statements of a certain form (formally, Σ^B_0 formulas), the theory $\mathbf{VTC}^0$ characterizes (uniform) polynomial-size TC^0-Frege proofs.

The construction of polynomial-size TC^0-Frege refutations for random 3CNF formulas, will consist of the following steps:

I. Formalize the following statement as a first-order formula:

$$\forall \text{ assignment } A \ \big(\mathbf{C} \text{ is a 3CNF and } w \text{ is its FKO unsatisfiabiliy witness} \longrightarrow \text{exists a clause } C_i \text{ in } \mathbf{C} \text{ such that } C_i(A) = 0\big), \tag{A.1}$$

where an *FKO witness* is a suitable formalization of the unsatisfiability witness defined by Feige, Kim and Ofek [37]. The corresponding predicate is called *the FKO predicate*.

II. Prove formula (A.1) in the theory $\mathbf{VTC}^0$.

III. Translate the proof in Step II into a family of propositional TC^0-Frege proofs (of the family of propositional translations of (A.1)). By Theorem A.42 (proved in [29]), this will be a polynomial-size propositional proof (in the size of $\mathbf{C}$). The translation of (A.1) will consist of a family of propositional formulas of the form:

$$[\![\mathbf{C} \text{ is a 3CNF and } w \text{ is its FKO unsatisfiabiliy witness}]\!] \longrightarrow [\![\text{exists a clause } C_i \text{ in } \mathbf{C} \text{ such that } C_i(A) = 0]\!], \tag{A.2}$$

where $[\![\cdot]\!]$ denotes the mapping from first-order formulas to families of propositional formulas. By the nature of the propositional translation (second-sort) variables in the original first-order formula translate into a collection of propositional variables. Thus, (A.2) will consist of propositional variables derived from the variables in (A.1).

IV. For the next step we first notice the following two facts:

(i) Assume that $\underline{\mathbf{C}}$ is a random 3CNF with n variables and $cn^{1.4}$ clauses (for a sufficiently large constant c). By [37], with high probability there exists an FKO unsatisfiability witness $\underline{w}$ for $\underline{\mathbf{C}}$. Both $\underline{w}$ and $\underline{\mathbf{C}}$ can be encoded as finite sets of numbers, as required by the predicate

for 3CNF and the FKO predicate in (A.1). Let us identify $\underline{w}$ and $\underline{\mathbf{C}}$ with their encodings. Then, assuming (A.1) was formalized correctly, assigning $\underline{w}$ and $\underline{\mathbf{C}}$ to (A.1) satisfies the *premise* of the implication in (A.1).

(ii) Now, by the definition of the translation from first-order formulas to propositional formulas, if an object α satisfies the predicate $P(X)$ (i.e., $P(\alpha)$ is true in the standard model), then there is a propositional assignment of $0, 1$ values that satisfies the propositional translation of $P(X)$. Thus, by Item (i) above, there exists an $0, 1$ assignment ζ that satisfies the premise of (A.2) (i.e., the propositional translation of the premise of the implication in (A.1)).

In the current step we show that after assigning ζ to the conclusion of (A.2) (i.e., to the propositional translation of the conclusion in (A.1)) one obtains precisely $\neg\underline{\mathbf{C}}$ (formally, a renaming of $\neg\underline{\mathbf{C}}$, where $\neg\underline{\mathbf{C}}$ is the 3DNF obtained by negating $\underline{\mathbf{C}}$ and using the de Morgan laws).

V. Take the propositional proof obtained in (**III**), and apply the assignment ζ to it. The proof then becomes a polynomial-size TC^0-Frege proof of a formula $\phi \rightarrow \neg\underline{\mathbf{C}}$, where ϕ is a propositional sentence (without variables) logically equivalent to TRUE (because ζ satisfies it, by (**IV**)). From this, one can easily obtain a polynomial-size TC^0-Frege refutation of $\underline{\mathbf{C}}$ (or equivalently, a proof of $\neg\underline{\mathbf{C}}$).

The bulk of our work lies in (**I**) and especially in (**II**). We need to formalize the necessary properties used in proving the correctness of the FKO witnesses and show that the correctness argument can be carried out in the weak theory. There are two main obstacles in this process. The first obstacle is that the correctness (soundness) of the witness is originally proved using spectral methods, which assumes that eigenvalues and eigenvectors are over the *reals*; whereas the reals are not defined in our weak theory. The second obstacle is that one needs to prove the correctness of the witness, and in particular the part related to the spectral method, *constructively* (formally in our case, inside $\mathbf{VTC}^0$). Specifically, linear algebra is not known to be (computationally) in TC^0, and (proof-complexity-wise) it is conjectured that TC^0-Frege do not admit short proofs of the statements of linear algebra (more specifically still, short proofs relating to inverse matrices and the determinant properties; see [89] on this).

The first obstacle is solved using rational approximations of sufficient accuracy (polynomially small errors), and showing how to carry out the proof in the theory with such approximations. The second obstacle is solved basically by constructing the argument (the main formula above) in a way that exploits nondeterminism (i.e., in a way that enables supplying additional witnesses for the properties needed to prove the correctness of the original witness; e.g, all eigenvectors and all eigenvalues of the appropriate matrices in the original witness). In other words, we do not have to construct certain objects but can provide them, given the possibility to certify the property we need. Formally, this means that we put additional witnesses in the FKO predicate occurring in the main formula in (**I**) above.

A.0.6 Overview of the Proof

Preliminary notations. A *random* $3CNF$ is generated by choosing independently, with repetitions, m clauses with three literals each, out of all possible $2^3 \cdot \binom{n}{3}$ clauses with n variables $x_1, \ldots, x_n$. We say that a property holds *with high probability* when the probability is $1 - o(1)$. We now define the notion of TC^0 formulas and TC^0-Frege proofs. The system we give is only one of many possibilities to define such proof systems (see e.g. [18]). The class of TC^0 formulas are built up using unbounded fan-in connectives $\wedge, \vee, \neg$ and threshold gates Th_i, for $i \in \mathbb{N}$, where $\mathsf{Th}_i(A_1, \ldots, A_n)$ is true if and only if at least i of the A_k's are true. The *depth* of a formula is the maximal nesting of connectives in it and the *size* of a formula is the total number of connectives in it.

Definition A.1 (TC^0-Frege (informal))**.** A TC^0-Frege proof system is a sequent calculus with a set of standard sound derivation rules and axioms. We give only the following rule as an example:
($\mathsf{Th_i}$-left): From the sequents $\mathsf{Th}_i(A_2 \ldots A_n), \Gamma \longrightarrow \Delta$ and
$\mathsf{Th}_{i-1}(A_2 \ldots A_n), A_1, \Gamma \longrightarrow \Delta$ we may infer the sequent $\mathsf{Th}_i(A_1 \ldots A_n), \Gamma \longrightarrow \Delta$, for arbitrary TC^0 formulas A_i and sets Γ, Δ of TC^0 formulas. (The intended meaning of $\Gamma \longrightarrow \Delta$ is that the conjunction of the formulas in Γ implies the disjunction of the formulas in Δ.)

A TC^0-frege proof of a formula φ is a sequence of sequents $\pi = (S_1, \ldots, S_k)$ such that $S_k = \varphi$ and every sequent in it is either an axiom or was derived from previous lines by a derivation rule. The *size* of the proof π is the total size of all formulas in its sequents. The *depth* of the proof π is the maximal depth of a formula in its sequents. A TC^0-Frege proof of *a family of formulas* $\{\varphi_i : i \in \mathbb{N}\}$ is a family of sequences $\{(S^i_1, \ldots, S^i_{k^i}) : i \in \mathbb{N}\}$, where each S^i_j is a TC^0 formula that can be derived from some S^i_k for $k < j$ using the above rules, such that $S^i_{k^i} = \longrightarrow \varphi_i$, and there is a *common constant c bounding the depth of every formula in all the sequences.*

Overview of theories of bounded arithmetic. Here, we highlight the theories $\mathbf{VTC}^0$, as defined by Cook and Nguyen [29]. This is a (first-order) two-sorted theory, having a first sort for natural number variables and a second sort for bit strings (formally, they are finite sets of natural numbers whose characteristic vectors are bit strings). The theory $\mathbf{VTC}^0$ is not a propositional proof system but it corresponds to TC^0-Frege (Theorem A.42). We choose to work with the theory because it is easier to carry out proofs in a first-order theory than with propositional proofs.

The language of two-sorted arithmetic, denoted $\mathcal{L}^2_A$, consists of the following relation, function and constant symbols: $\{+, \cdot, \leq, 0, 1, |\ |, =_1, =_2, \in\}$. The intended semantic of this language is the standard model $\mathbb{N}_2$ of two-sorted arithmetic consisting of a first-sort universe $U_1 = \mathbb{N}$ and a second-sort universe U_2 of all finite subsets of $\mathbb{N}$. 0 and 1 are interpreted in $\mathbb{N}_2$ as zero and one. The functions $+$ and $\cdot$ are addition and multiplication of numbers, $\leq$ is the less-than relation on numbers. The function $|\ |$ maps a finite set of numbers to its largest element plus one. The relation $=_1$ is interpreted as equality between numbers, $=_2$ is interpreted as equality between finite sets of numbers. The relation $\in$ holds for a number n and a finite set of numbers N if and only if n is an element of

N. We denote the first-sort (number) variables by lower-case letters $x, y, z, ...$, and the second-sort (string) variables by capital letters $X, Y, Z, ...$ We can build formulas in the usual way, using two sorts of quantifiers, number quantifiers and string quantifiers. A number quantifier $\exists x$ ($\forall x$) is polynomially bounded if it is of the form $\exists x(x \leq f(n) \wedge \dots)$ ($\forall x(x \leq f(n) \rightarrow \dots)$) for some number term f. Given some function symbol f, a formula φ is in $\Sigma_0^B(f)$ if it uses no string quantifiers and all number quantifiers are polynomially bounded and it possibly uses the function symbol f. We represent a finite set of natural numbers N by a finite string $S_N = S_N^0 \dots S_N^{|N|-1}$ such that $S_N^i = 1$ if and only if $i \in N$. We will abuse notation and identify N and S_N.

We now state informally the formalizations of the proofs in the theory, rather. For this overview it is sufficient to know only two things about $\mathbf{VTC}^0$. First, the kind of reasoning (formally, proofs) the theory allows: the theory $\mathbf{VTC}^0$ is meant to allow reasoning that involves counting. Specifically, it enables one to use the function $numones(X)$ whose value is the number of ones in the string X (or equivalently, the number of elements in the set X). Second, the relation between $\mathbf{VTC}^0$ and TC^0-Frege, described below.

Definition A.2 (Propositional translation (informal))**.** Let $\varphi(\vec{x}, \vec{X})$ be a Σ_0^B formula. The *propositional translation* of φ is a family $[\![\varphi]\!] = \{[\![\varphi]\!]_{\vec{m};\vec{n}} \mid m_i, n_i \in \mathbb{N}\}$ of propositional formulas in variables $p_j^{X_i}$ for every $X_i \in \vec{X}$. The intended meaning is that $[\![\varphi]\!]$ is a valid family of formulas if and only if the formula

$$\forall \vec{x} \forall \vec{X} (\bigwedge |X_i| = \underline{n_i}) \rightarrow \varphi(\underline{\vec{m}}, \vec{X})$$

is true in the standard model $\mathbb{N}_2$ of two sorted arithmetic. For given $\vec{m}, \vec{n} \in \mathbb{N}$ we will define $[\![\varphi]\!]$ by induction on the complexity of the formula $[\![\varphi]\!]_{\vec{m};\vec{n}}$.

We can now state the relation between provability of an arithmetical statement φ in $\mathbf{VTC}^0$ to the provability of the family $[\![\varphi]\!]$ in TC^0-Frege as follows.

Theorem A.3 (Section X.4.3. [29])**.** *Let $\varphi(\vec{x}, \vec{X})$ be a Σ_0^B formula. Then, if $\mathbf{VTC}^0$ proves $\varphi(\vec{x}, \vec{X})$ then there is a polynomial size family of TC^0-Frege proofs of $[\![\varphi]\!]$.*

Feige-Kim-Ofek witnesses and the main formula. We now sketch the main two-sorted first-order formula we are going to prove in the theory $\mathbf{VTC}^0$. Basically, it will formalize the correctness of the Feige et al. witnesses. The formula we construct will itself speak about 3CNFs $\mathbf{C} = \bigwedge_{\alpha=1}^{m} C_\alpha$, where n is the number of variables and m is the number of clauses. Each clause C_α is of the form $x_i^{\ell_1} \vee x_j^{\ell_2} \vee x_k^{\ell_3}$, for $\ell_1, \ell_2, \ell_3 \in \{0, 1\}$, where x_i^1 abbreviates x_i and x_i^0 abbreviates $\neg x_i$. A clause C_α is coded by the sequence $\langle i, j, k, \langle \ell_1, \ell_2, \ell_3 \rangle, \alpha \rangle$, where $\langle \cdot, \dots, \cdot \rangle$ is the tupling function and $\langle \cdot, \dots, \cdot \rangle_j^k$ is the jth element in a k-tuple. The defining Σ_0^B-formula of the relation is:

$$\begin{aligned} \textsc{Clause}(x, n, m) \leftrightarrow & \exists i, j, k \leq n \exists \alpha \leq m \exists \ell \leq 8. \\ & \langle x \rangle_1^5 = i \wedge \langle x \rangle_2^5 = j \wedge \langle x \rangle_3^5 = k \wedge \langle x \rangle_4^5 = \ell \wedge \langle x \rangle_5^5 = \alpha. \end{aligned}$$

For some clause C and a string variable A (interpreted as a Boolean assignment), we define by a $\Sigma_0^B(numones)$ formula the following predicate, stating that

C is not satisfied under the assignment A:

$$\begin{aligned}\textsc{NotSAT}(C, A) \equiv \exists i, j, k \leq n \big(&\langle C\rangle_1^5 = i \wedge (A(i) \leftrightarrow \langle\langle C\rangle_4^5\rangle_1^3 = 0)\big) \\ &\wedge \big(\langle C\rangle_2^5 = j \wedge (A(j) \leftrightarrow \langle\langle C\rangle_4^5\rangle_2^3 = 0)\big) \\ &\wedge \big(\langle C\rangle_3^5 = k \wedge (A(k) \leftrightarrow \langle\langle C\rangle_4^5\rangle_3^3 = 0)\big).\end{aligned}$$

The *imbalance* of a variable x_i is the absolute value of its positive occurrences and negative occurrences in $\mathbf{C}$. The *imbalance of* $\mathbf{C}$ is the sum over the imbalances of all variables (whose predicate is denoted $\textsc{Imb}(\mathbf{C}, I)$).

We now state the main formula that we are going to prove in $\mathbf{VTC}^0$. It says that if there exists a certain witness with certain properties then there exists a clause in $\mathbf{C}$ that is not satisfied by any assignment A (one can think of all the free variables in the formula as universally quantified). To actually construct the main formula, we need to define several predicates that we do not have the space to cover in this extended abstract. One important predicate is $\textsc{EigValBound}(M, \vec{\lambda}, V)$ that ensures that given the matrix M, which will correspond to the 3CNF $\mathbf{C}$ in a manner made precise (via the $\textsc{Mat}(M, \mathbf{C})$ predicate), $\vec{\lambda}$ is a collection of n rational approximations of the normalized eigenvalues of M and that V is the rational matrix whose rows are the rational approximations of the eigenvectors of M. Another predicate is the $\textsc{Coll}(t, k, d, n, m, \mathbf{C}, \mathscr{D})$ that states that $\mathscr{D}$ is a (t, k, d)-*collection* of t inconsistent k-tuples of $\mathbf{C}$ (with up to d intersections)—which is a concept introduced in [37].

Definition A.4 (Main Formula)**.** The *Main Formula* is the following $\Sigma_0^B(\mathit{numones})$ formula ($\vec{\lambda}$ denotes n distinct number parameters $\lambda_1, \ldots, \lambda_n$):

$$\Bigg(\textsc{3CNF}(\mathbf{C}, n, m) \wedge \textsc{Coll}(t, k, d, n, m, \mathbf{C}, \mathscr{D}) \wedge \textsc{Imb}(\mathbf{C}, I) \wedge \textsc{Mat}(M, \mathbf{C}) \wedge$$

$$\textsc{EigValBound}(M, \vec{\lambda}, V) \wedge \lambda = \max\{\vec{\lambda}\} \wedge t > \frac{d \cdot (I + \lambda n)}{2} + o(1)\Bigg)$$

$$\longrightarrow \exists i \leq m\, \textsc{NotSAT}(\mathbf{C}[i], A).$$

Proof of the main formula. The following is our key theorem:

Theorem A.5 (Key)**.** *The theory* $\mathbf{VTC}^0$ *proves the Main Formula (Definition A.48).*

The proof in the theory follows the proof of correctness of the unsatisfiability witnesses introduced in Feige et al. [37]. Showing how to carry out this proof in $\mathbf{VTC}^0$ constitutes our main body of work. We shall only state some of the lemmas we prove, and discuss the spectral bound. The proofs can be found in the attached full version.

Proof. (Overview) We reason inside $\mathbf{VTC}^0$. Assume by a way of contradiction that the premise of the implication in the Main Formula holds and that there is an assignment $A \in \{0, 1\}^n$ (construed as a string variable of length n) that satisfies every clause in $\mathbf{C}$. We define the following sets and functions in the theory. Let

$\mathsf{satLit}(A, \mathbf{C})$ be the string function that outputs the set of all positions of literals in $\mathbf{C}$ that are satisfied by A. If the literals of a clause are not all true or not all false under A, then we say that the clause is satisfied as NAE (standing for "not all equal") by A. Let $\mathsf{satNAE}(A, \mathbf{C})$ be the string function that returns the set of all clauses in $\mathbf{C}$ that are satisfied as NAE by A.

Lemma A.6. $\mathbf{VTC}^0$ *proves:* $\mathit{numones}(\mathsf{satLit}(A, \mathbf{C})) \leq \frac{3m+I}{2}$.

We now bound the number of clauses in $\mathbf{C}$ that contain exactly two literals satisfied by A.

Lemma A.7. $\mathbf{VTC}^0$ *proves: Assume the premise of the Main Formula and let h be the number of clauses in $\mathbf{C}$ that contain exactly two literals satisfied by A. Then*

$$h \leq \frac{3m+I}{2} - 3m + 2 \cdot \mathit{numones}(\mathsf{satNAE}(A, \mathbf{C}))\,.$$

The following lemma provides an upper bound on the number of clauses in $\mathbf{C}$ that can be satisfied as NAE by the assignment A.

Lemma A.8. *(Assuming the premise of the Main Formula)* $\mathbf{VTC}^0$ *proves that:*

$$\mathit{numones}(\mathsf{satNAE}(A, \mathbf{C})) \leq (n\lambda + 3m)/4 + o(1).$$

The proof of this lemma involves a *spectral argument*. Carrying out this argument in the theory is fairly difficult because one has to cope with rational approximations (as the eigenvalues and eigenvectors might be irrationals, and so undefined in the theory) and further the proof must be sufficiently constructive, in the sense that it would fit in the theory $\mathbf{VTC}^0$. Specifically, we need the following lemma (for an assignment A we define its associated vector $\mathbf{a} \in \{-1, 1\}^n$ such that $\mathbf{a}(i) = 1$ if $A(i) = 1$ and $\mathbf{a}(i) = -1$ if $A(i) = 0$):

Lemma A.9 (Main spectral bound)**.** *The theory* $\mathbf{VTC}^0$ *proves: assuming* $\textsc{EigValBound}(M, \vec{\lambda}, V)$ *holds, then for any assignment A to n variables:*

$$\mathbf{a}^t M \mathbf{a} \leq \lambda n + o(1)\,. \tag{A.3}$$

The idea of proving the spectral bound in $\mathbf{VTC}^0$ (Lemma A.9). We explain here informally how we proceed to prove the bound $\mathbf{a}^t M \mathbf{a} \leq \lambda n + o(1)$, for any $\mathbf{a} \in \{-1, 1\}^n$, in the theory $\mathbf{VTC}^0$, assuming that $\textsc{Mat}(M, \mathbf{C})$ and $\textsc{EigValBound}(M, \vec{\lambda}, V)$ hold (Lemma A.68). The idea of the proof of this inequality is as follows: in the predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$ we certify that the rows of a given matrix V are rational approximations of the normalized eigenvector basis of M. Since M is symmetric and real, V will approximate an orthonormal matrix, and V^t will *approximate* V^{-1} (this is where we circumvent the need to prove the correctness of inverting a matrix in the theory $\mathbf{VTC}^0$). Thus, V^{-1} approximates the matrix of the basis transformation from the standard basis to the eigenvector basis. Note that $\mathbf{a}$ (as a $\{-1, 1\}$ vector) is already almost described in the standard basis. Hence, it will be possible to prove in the theory that $V^t\mathbf{a}$ is the representation of $\mathbf{a}$ in the (approximate) eigenvector basis, i.e., we shall have an equality $\mathbf{a} = \sum_{i=1}^{n} \gamma_i \mathbf{v}_i + o(1)$, for $\mathbf{v}_i$'s the approximate eigenvectors of M and rationals γ_i's. After plugging-in this equality in $\mathbf{a}^t M \mathbf{a}$, to prove $\mathbf{a}^t M \mathbf{a} \leq \lambda n$ we only need to *validate computations*—using also the fact

that we know the inequalities $M\mathbf{v}_i \leq \lambda \mathbf{v}_i + o(1)$, for any $i \in [n]$ (since this will be witnessed in the predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$ as well).

We can now finish the proof of the key theorem. In $\mathbf{VTC}^0$ (and assuming the premise of the Main Formula), let h be the number of clauses in $\mathbf{C}$ that contain exactly two literals satisfied by A. We have:

$$\begin{aligned} h &\leq \frac{3m+I}{2} - 3m + 2 \cdot \mathit{numones}(\mathsf{satNAE}(A, \mathbf{C})) && \text{(by Lemma A.52)} \\ &\leq \frac{3m+I}{2} - 3m + \frac{3m+\lambda n}{2} + o(1) = \frac{I+\lambda n}{2} + o(1)\,. && \text{(by Lemma A.55)} \end{aligned} \tag{A.4}$$

Since we assumed that A satisfies $\mathbf{C}$, then every clause in $\mathbf{C}$ has at least one literal satisfied by A. Thus, the clauses in $\mathbf{C}$ that are not satisfied as 3XOR by A are precisely the clauses that have exactly two literals satisfied by A. By (A.36), the number of clauses that have exactly two literals satisfied by A is at most $\frac{I+\lambda n}{2} + o(1)$. We now use our witness, assumed to exist in the premise of the Main Formula, to show that:

Lemma A.10 (In $\mathbf{VTC}^0$)**.** *(Assuming the premise of the Main Formula) the number of clauses in* $\mathbf{C}$ *that are not satisfied as 3XOR by A is at least* $\lceil t/d \rceil$*.*

Thus, by Lemma A.10 and the fact that the number of clauses in $\mathbf{C}$ not satisfied as 3XOR by A is at most $\frac{I+\lambda n}{2} + o(1)$, we get $t = d \cdot \frac{t}{d} \leq d \cdot \left\lceil \frac{t}{d} \right\rceil \leq d \cdot \frac{I+\lambda n}{2} + o(1)$, which contradicts our assumption (in the Main Formula) that $t > \frac{d(I+\lambda n)}{2} + o(1)$. □

A.0.7 Organization of the paper

The remainder of the paper is organized as follows. Section A.1 contains general preliminary definitions and notations, including propositional proof systems and the TC^0-Frege proof system. Section A.2 contains a long exposition of the basic logical setting we use, that is, the relevant theories of (two-sorted) bounded arithmetic ($\mathbf{V}^0$ and $\mathbf{VTC}^0$, from [29]), and a detailed explanation of how to formalize certain proofs in these theories. This includes defining certain syntactic objects in the theories as well as counting and doing computations in the theory. Readers who already know the basics of bounded arithmetic can skip Section A.2, and look only at specific parts or definitions, when needed. Section A.3 provides the formalization of the main formula we prove in the theory. This formula expresses the correctness of the Feige at al. witnesses for unsatisfiability [37]. Section A.4 contains the proof of the main formula, excluding the lemma establishing the spectral inequality which is deferred to a section of its own. Section A.5 provides the full proof in the theory of the spectral inequality. Section A.6 finally puts everything together, and shows how to obtain short propositional refutations from the proof in the theory of the main formula.

A.1 Preliminaries

We write $[n]$ for $\{1, \ldots, n\}$. We denote by $\top, \bot$ the truth values *true* and *false*, respectively.

Definition A.11 (3CNF)**.** A *literal* is a propositional variable x_i or its negation $\neg x_i$. A 3-*clause* is a disjunction of three literals. A 3CNF is a conjunction of 3-clauses.

Definition A.12 (Random 3CNF)**.** A *random 3CNF* is generated by choosing independently, with repetitions, m clauses with three literals each, out of all possible $2^3 \cdot \binom{n}{3}$ clauses with n variables $x_1, \ldots, x_n$.

We say that a property holds *with high probability* when it holds with probability $1 - o(1)$.

A.1.1 Miscellaneous linear algebra notations

We denote by $\mathbb{R}^k$ and $\mathbb{Q}^k$ the k-dimensional real and rational vector spaces in the canonical basis $e_1, \ldots, e_k$. The vectors in these spaces are given as sequences $a = (a_1 \ldots a_k)$. In this context for some k-dimensional vector space V and two vectors $a, b \in V$ by $\langle a, b \rangle$ we denote the *inner product* of a and b which is defined by $\langle a, b \rangle := \sum_{i=1}^{k} a_i \cdot b_i$. Two vectors a, b are *orthogonal* if $\langle a, b \rangle = 0$. The *(Euclidean) norm* of a vector a is denoted by $||a||$ and is defined as $\sqrt{\sum_{i=1}^{k} a_i^2}$. A vector a is called *normal* if $||a|| = 1$. A set of vectors is called *orthonormal* if they are pairwise orthogonal and normal. A function $f : V \longrightarrow W$ is *linear* if for all $v, w \in V$, $f(c_1 v + c_2 w) = c_1 f(v) + c_2 f(w)$. Every linear function $f : V \longrightarrow W$ can be represented by a matrix $A_f = (a_{i,j})_{i \leq dim(W), j \leq dim(V)}$. Observe that the representation depends not only on f but also on the bases of V and W. A matrix $A = (a_{i,j})$ is symmetric if $a_{i,j} = a_{j,i}$ for all i, j. If for some matrix A and vector v it holds that $Av = \lambda v$ we call v an *eigenvector* and λ an *eigenvalue* of A.

Fact 1 (cf. [53])**.** The eigenvectors of any real symmetric matrix $A : V \longrightarrow V$ are an orthogonal basis of V, and the eigenvalues of A are all real numbers.

A.1.2 Propositional proofs and TC^0-Frege systems

In this section we define the notion of TC^0 formulas. Then we define the propositional proof system TC^0-Frege as a sequent calculus operating with TC^0 formulas and prove basic properties of it. We will follow the exposition from [29]. The system we give is only one of many possibilities to define such proof systems (see e.g. [18] for a polynomially-equivalent definition).

The class of TC^0 formulas consists basically of unbounded fan-in constant depth formulas with $\wedge, \vee, \neg$ and threshold gates. Formally, we define:

Definition A.13 (TC^0 formula)**.** A TC^0 formula is built from

(i) propositional constants $\bot$ and $\top$,

(ii) propositional variables p_i for $i \in \mathbb{N}$,

(iii) connectives $\neg$ and Th_i, for $i \in \mathbb{N}$.

Items (i) and (ii) constitute the *atomic formulas*. TC^0 formulas are defined inductively from atomic formulas via the connectives:

(a) if A is a formula, then so is $\neg A$ and

(b) for $n > 1$ and $i \in \mathbb{N}$, if $A_1, \ldots, A_n$ are formulas, then so is $\mathsf{Th}_i A_1 \ldots A_n$.

The *depth* of a formula is the maximal nesting of connectives in it and the *size* of the formula is the total number of connectives in it.

For the sake of readability we will also use parentheses in our formulas, though they are not necessary. The semantics of the *Threshold Connectives* Th_i are as follows. $\mathsf{Th}_i(A_1, \ldots, A_n)$ is true if and only if at least i of the A_k are true. Therefore we will abbreviate $\mathsf{Th}_i(A_1, \ldots, A_i)$ as $\bigwedge_{k \leq i} A_k$ and $\mathsf{Th}_1(A_1, \ldots, A_i)$ as $\bigvee_{k \leq i} A_k$. Moreover we let $\mathsf{Th}_0(A_1, \ldots, A_n) = \top$ and $\mathsf{Th}_i(A_1, \ldots, A_n) = \bot$, for $i > n$.

The following is the sequent calculus TC^0-Frege.

Definition A.14 (TC^0-Frege)**.** A TC^0-Frege proof system is a sequent calculus with the axioms

$$A \longrightarrow A, \qquad \bot \longrightarrow, \qquad \longrightarrow \top,$$

where A is any TC^0 formula, and the following derivation rules:

Weaken-left: From the sequent $\Gamma \longrightarrow \Delta$ we may infer the sequent $\Gamma, A \longrightarrow \Delta$.

Weaken-right: From the sequent $\Gamma \longrightarrow \Delta$ we may infer the sequent $\Gamma \longrightarrow A, \Delta$.

Exchange-left: From the sequent $\Gamma_1, A_1, A_2, \Gamma_2 \longrightarrow \Delta$ we may infer the sequent $\Gamma_1, A_2, A_1, \Gamma_2 \longrightarrow \Delta$.

Exchange-right: From the sequent $\Gamma \longrightarrow \Delta_1, A_1, A_2, \Delta_2$ we may infer the sequent $\Gamma \longrightarrow \Delta_1, A_2, A_1, \Delta_2$.

Contract-left: From the sequent $\Gamma, A, A \longrightarrow \Delta$ we may infer the sequent $\Gamma, A \longrightarrow \Delta$.

Contract-right: From the sequent $\Gamma \longrightarrow A, A, \Delta$ we may infer the sequent $\Gamma \longrightarrow A, \Delta$.

$\neg$-left: From the sequent $\Gamma \longrightarrow A, \Delta$ we may infer the sequent $\Gamma, \neg A \longrightarrow \Delta$.

$\neg$-right: From the sequent $\Gamma, A \longrightarrow \Delta$ we may infer the sequent $\Gamma \longrightarrow \neg A, \Delta$.

All-left: From the sequent $A_1, \ldots, A_n, \Gamma \longrightarrow \Delta$ we may infer the sequent $\mathsf{Th}_n A_1 \ldots A_n, \Gamma \longrightarrow \Delta$.

All-right: From the sequents $\Gamma \longrightarrow A_1, \Delta$, $\ldots$, $\Gamma \longrightarrow A_n, \Delta$ we may infer the sequent $\Gamma \longrightarrow \mathsf{Th}_n A_1 \ldots A_n, \Delta$.

One-left: From the sequents $A_1, \Gamma \longrightarrow \Delta$, $\ldots$, $A_1, \Gamma \longrightarrow \Delta$ we may infer the sequent $\mathsf{Th}_1 A_1 \ldots A_n, \Gamma \longrightarrow \Delta$.

One-right: From the sequent $\Gamma \longrightarrow A_1, \ldots, A_n, \Delta$ we may infer the sequent $\Gamma \longrightarrow \mathsf{Th}_1 A_1 \ldots A_n, \Delta$.

Th_i-left: From the sequents $\mathsf{Th}_i A_2 \ldots A_n, \Gamma \longrightarrow \Delta$ and $\mathsf{Th}_{i-1} A_2 \ldots A_n, A_1, \Gamma \longrightarrow \Delta$ we may infer the sequent $\mathsf{Th}_i A_1 \ldots A_n, \Gamma \longrightarrow \Delta$.

Th_i-right: From sequents $\Gamma \longrightarrow \mathsf{Th}_i A_2 \ldots A_n, A_1, \Delta$ and $\Gamma \longrightarrow \mathsf{Th}_{i-1} A_2 \ldots A_n, \Delta$ we may infer the sequent $\Gamma \longrightarrow \mathsf{Th}_i A_1 \ldots A_n, \Delta$.

Cut: From the sequents $\Gamma \longrightarrow A, \Delta$ and $\Gamma, A \longrightarrow \Delta$ we may infer the sequent $\Gamma \longrightarrow \Delta$,

for arbitrary TC^0 formulas A_i and sets Γ, Δ of TC^0 formulas. The intended meaning of $\Gamma \longrightarrow \Delta$ is that the conjunction of the formulas in Γ implies the disjunction of the formulas in Δ. A TC^0-frege proof of a formula φ is a sequence of sequents $\pi = (S_1, \ldots, S_k)$ such that $S_k = \longrightarrow \varphi$ and every sequent in it is either an axiom or was derived from previous lines by a derivation rule. The *size* of the proof π is the total size of all formulas in its sequents. The *depth* of the proof π is the maximal depth of a formula in its sequents. A TC^0-Frege proof of *a family of formulas* $\{\varphi_i : i \in \mathbb{N}\}$ is a family of sequences $\{(S_1^i, \ldots, S_{k^i}^i) : i \in \mathbb{N}\}$, where each S_j^i is a TC^0 formula that can be derived from some S_k^i for $k < j$ using the above rules, such that $S_{k^i}^i = \longrightarrow \varphi_i$, and there is a *common constant c bounding the depth of every formula in all the sequences.*

Proposition A.15. The proof system TC^0-Frege is sound and complete. That is, every formula A proven in the above way is a tautology and every tautology can be derived by proofs in the above sense.

Definition A.16 (Polynomial simulation; separation)**.** Let P, Q be two propositional proof systems that establish Boolean tautologies (or refute unsatisfiable Boolean formulas, or refute unsatisfiable CNF formulas). We say that P *polynomially simulates* Q if there is a polynomial-time computable function f such that given a Q-proof of τ outputs a P-proof of τ. If P does not polynomially simulate Q or vice versa we say that P is *separated* from Q.

Sometimes, it is enough to talk about *weak* polynomial simulations: we say that a proof system P *weakly polynomially simulates* the proof system Q if there is a polynomial p such that for every propositional tautology τ, if the minimal Q-proof of τ is of size s then the minimal P-proof of τ is of size at most $p(s)$.

For a possibly partial $\{0,1\}$ assignment $\vec{a}$ to the propositional variables, we write $\varphi[\vec{a}]$ to denote the formula φ in which propositional variables are substituted by their values in $\vec{a}$. For a proof $\pi = (\varphi_1, \ldots, \varphi_\ell)$ we write $\pi[\vec{a}]$ to denote $\pi = (\varphi_1[\vec{a}], \ldots, \varphi_\ell[\vec{a}])$. The system TC^0-Frege can efficiently evaluate assignments to some of the variables of formulas in the following sense.

Claim A.17. Let $\varphi(\vec{p}, \vec{q})$ be a propositional formula in variables $p_1 \ldots p_{m_1}$ and $q_1 \ldots q_{m_2}$ and let $\vec{a} \in \{0,1\}^{m_1}$. If TC^0-Frege proves $\varphi(\vec{p}, \vec{q})$ with a proof π_φ of length n, then it also proves $\varphi(\vec{a}, \vec{q})$ in a proof $\pi_{\varphi[\vec{a}]}$ of length n. Additionally, for any formula $\varphi(\vec{p})$ in variables $p_1 \ldots p_{m_1}$ and an assignment $\vec{a} \in \{0,1\}^{m_1}$, TC^0-Frege has polynomial size proofs of either $\varphi[\vec{a}]$ or $\neg\varphi[\vec{a}]$.

Proof sketch: Consider with π_φ and substitute each occurrence of p_i by a_i. The resulting proof remains correct and proves $\varphi(\vec{a}, \vec{q})$, because every TC^0-Frege rule application is still correct after the assignment.

The second claim is proved by induction over the complexity of φ. If $\varphi[\vec{a}]$ is true we can construct a proof by proving the (substitution instances of the) atomic formulas and then proceeding using the appropriate rules of the calculus by the way the formula is built up.

If $\varphi[\vec{a}]$ is false, then we proceed in the same way as above with $\neg\varphi[\vec{a}]$ instead of $\varphi[\vec{a}]$. $\blacksquare_{\text{Claim}}$

A.2 Theories of Bounded Arithmetic

In this section we give some of the necessary background from logic. Specifically, we present the theory $\mathbf{V}^0$ and its extension $\mathbf{VTC}^0$, as developed by Cook and Nguyen [29] (see also [92]). These are weak systems of arithmetic, namely, fragments of Peano Arithmetic, usually referred to as theories of Bounded Arithmetic (for other treatments of theories of bounded arithmetic see also [20, 49, 57]). The theories are (first-order) two-sorted theories, having a first sort for natural numbers and a second sort for finite sets of numbers (representing bit-strings via their characteristic functions). The theory $\mathbf{V}^0$ corresponds (in a manner made precise) to bounded depth Frege, and $\mathbf{VTC}^0$ corresponds to TC^0-Frege (see Section A.2.2). The complexity classes $\mathbf{AC}^0$, TC^0, and their corresponding function classes $\mathbf{FAC}^0$ and $\mathbf{FTC}^0$ are also defined using the two-sorted universe (specifically, the first-ordered sort [numbers] are given to the machines in unary representation and the second-sort as binary strings).

Definition A.18 (Language of two-sorted arithmetic $\mathcal{L}_A^2$)**.** The language of two-sorted arithmetic, denoted $\mathcal{L}_A^2$, consists of the following relation, function and constant symbols:

$$\{+, \cdot, \leq, 0, 1, |\ |, =_1, =_2, \in\}\,.$$

We describe the intended meaning of the symbols by considering the standard model $\mathbb{N}_2$ of two-sorted Peano Arithmetic. It consists of a first-sort universe $U_1 = \mathbb{N}$ and a second-sort universe U_2 of all finite subsets of $\mathbb{N}$. The constants 0 and 1 are interpreted in $\mathbb{N}_2$ as the appropriate natural numbers zero and one, respectively. The functions $+$ and $\cdot$ are the usual addition and multiplication on the universe of natural numbers, respectively. The relation $\leq$ is the appropriate "less or equal than" relation on the first-sort universe. The function $|\cdot|$ maps a finite set of numbers to its largest element plus one. The relation $=_1$ is interpreted as equality between numbers, $=_2$ is interpreted as equality between finite sets of numbers. The relation $n \in N$ holds for a number n and a finite set of numbers N if and only if n is an element of N.

We denote the first-sort (number) variables by lower-case letters $x, y, z, ...$, and the second-sort (string) variables by capital letters $X, Y, Z, ...$ We build formulas in the usual way, using two sorts of quantifiers: number quantifiers and string quantifiers. A number quantifier is said to be *bounded* if it is of the form $\exists x(x \leq t \wedge \dots)$ or $\forall x(x \leq t \rightarrow \dots)$, respectively, for some number term t that does not contain x. We abbreviate $\exists x(x \leq t \wedge \dots)$ and $\forall x(x \leq t \rightarrow \dots)$ by

$\exists x \leq t$ and $\forall x \leq t$, respectively. A string quantifier is said to be *bounded* if it is of the form $\exists X(|X| \leq t \wedge \dots)$ or $\forall X(|X| \leq t \rightarrow \dots)$ for some number term t that does not contain X. We abbreviate $\exists X(|X| \leq t \wedge \dots)$ and $\forall X(|X| \leq t \rightarrow \dots)$ by $\exists X \leq t$ and $\forall X \leq t$, respectively. A formula is in Σ^B_0 or Π^B_0 if it uses no string quantifiers and all number quantifiers are bounded. A formula is in Σ^B_{i+1} or Π^B_{i+1} if it is of the form $\exists X_1 \leq t_1 \dots \exists X_m \leq t_m \psi$ or $\forall X_1 \leq t_1 \dots \forall X_m \leq t_m \psi$, where $\psi \in \Pi^B_i$ and $\psi \in \Sigma^B_i$, respectively, and t_i does not contain X_i, for all $i = 1, \dots, m$. We write $\forall\Sigma^B_0$ to denote the universal closure of Σ^B_0. (i.e., the class of Σ^B_0-formulas that possibly have (not necessarily bounded) universal quantifiers in their front). We usually abbreviate $t \in T$, for a number term t and a string term T, as $T(t)$.

For a language $\mathcal{L} \supseteq \mathcal{L}^2_A$ we write $\Sigma^B_0(\mathcal{L})$ to denote Σ^B_0 formulas in the language $\mathcal{L}$.

As mentioned before a finite set of natural numbers N represents a finite string $S_N = S^0_N \dots S^{|N|-1}_N$ such that $S^i_N = 1$ if and only if $i \in N$. We will abuse notation and identify N and S_N.

In the context of a proof in the theory, we write n^c to mean the term $\underbrace{n \cdots n}_{c \text{ times}}$.

The (first-order) two-sorted proof system $\mathbf{LK}^2$. For proving statements in the two-sorted theories we need to specify a proof system to work with (this should not be confused with the propositional proof system we use). We shall work with a standard (two sorted) sequent calculus $\mathbf{LK}^2$ as defined in [29], section IV.4. This sequent calculus includes the standard logical rules of the sequent calculus for first-order logic **LK** augmented with four rules for introducing second-sort quantifiers. We also have the standard equality axioms (for first- and second-sorts) for the underlying language $\mathcal{L}^2_A$ (and when we extend the language, we assume we also add the equality axioms for the additional function and relation symbols). It is not essential to know precisely the system $\mathbf{LK}^2$ since we shall not be completely formal when proving statements in the two-sorted theories.

A.2.1 The theory $\mathbf{V}^0$

The base theory we shall work with is $\mathbf{V}^0$ and it consists of the following axioms:

Basic 1. $x + 1 \neq 0$ — **Basic 2**. $x + 1 = y + 1 \rightarrow x = y$

Basic 3. $x + 0 = x$ — **Basic 4**. $x + (y + 1) = (x + y) + 1$

Basic 5. $x \cdot 0 = 0$ — **Basic 6**. $x \cdot (y + 1) = (x \cdot y) + x$

Basic 7. $(x \leq y \wedge y \leq x) \rightarrow x = y$ — **Basic 8**. $x \leq x + y$

Basic 9. $0 \leq x$ — **Basic 10**. $x \leq y \vee y \leq x$

Basic 11. $x \leq y \leftrightarrow x < y + 1$ — **Basic 12**. $x \neq 0 \rightarrow \exists y \leq x(y + 1 = x)$

L1. $X(y) \rightarrow y < |X|$ — **L2**. $y + 1 = |X| \rightarrow X(y)$

SE. $(|X| = |Y| \wedge \forall i \leq |X| \, (X(i) \leftrightarrow Y(i))) \rightarrow X = Y$

Σ^B_0-COMP. $\exists X \leq y \forall z < y(X(z) \leftrightarrow \varphi(z))$, for all $\varphi \in \Sigma^B_0$ where X does not occur free in φ.

Here, the Axioms **Basic 1** through **Basic 12** are the usual axioms used to define Peano Arithmetic without induction (PA^-), which settle the basic properties of addition, multiplication, ordering, and of the constants 0 and 1. The Axiom **L1** says that the length of a string coding a finite set is an upper bound to the size of its elements. **L2** says that $|X|$ gives the largest element of X plus 1. **SE** is the extensionality axiom for strings which states that two strings are equal if they code the same sets. Finally, Σ_0^B**-COMP** is the comprehension axiom scheme for Σ_0^B formulas (it is an axiom for each such formula) and implies the existence of all sets which contain exactly the elements that fulfill any given Σ_0^B property.

When speaking about theories we will always assume that the theories are two-sorted theories.

Proposition A.19 (Corollary V.1.8. [29])**.** The theory $\mathbf{V}^0$ proves the (number) induction axiom scheme for Σ_0^B formulas Φ:

$$(\Phi(0) \wedge \forall x\,(\Phi(x) \rightarrow \Phi(x+1))) \rightarrow \forall z\,\Phi(z).$$

In the above induction axiom, x is a number variable and Φ can have additional free variables of both sorts.

The following is a basic notion needed to extend our language with new function symbols (we write $\exists! y\Phi$ to denote $\exists x(\Phi(x) \wedge \forall y(\Phi(y/x) \rightarrow x = y))$, where y is a new variable not appearing in Φ):

Definition A.20 (Two-sorted definability)**.** Let $\mathcal{T}$ be a theory over the language $\mathcal{L} \supseteq \mathcal{L}_A^2$ and let Φ be a set of formulas in the language $\mathcal{L}$. A number function f is Φ-definable in a theory $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, y, \vec{X})$ in Φ such that $\mathcal{T}$ proves

$$\forall\vec{x}\forall\vec{X}\exists! y\varphi(\vec{x}, y, \vec{X})$$

and it holds that[2]

$$y = f(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, y, \vec{X}). \tag{A.5}$$

A string function F is Φ-definable in a theory $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, \vec{X}, Y)$ in Φ such that $\mathcal{T}$ proves

$$\forall\vec{x}\forall\vec{X}\exists! Y\varphi(\vec{x}, \vec{X}, Y)$$

and it holds that

$$Y = F(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}, Y). \tag{A.6}$$

Finally, a relation $R(\vec{x}, \vec{X})$ is Φ-definable in a theory $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, \vec{X}, Y)$ in Φ such that it holds that

$$R(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}). \tag{A.7}$$

The formulas (A.5), (A.6), and (A.7) are the *defining axioms* for f, F, and R, respectively.

Definition A.21 (Conservative extension of a theory)**.** Let $\mathcal{T}$ be a theory in the language $\mathcal{L}$. We say that a theory $\mathcal{T}' \supseteq \mathcal{T}$ in the language $\mathcal{L}' \supseteq \mathcal{L}$ is *conservative over* $\mathcal{T}$ if every $\mathcal{L}$ formula provable in $\mathcal{T}'$ is also provable in $\mathcal{T}$.

[2]Meaning it holds in the standard two-sorted model $\mathbb{N}_2$.

We can expand the language $\mathcal{L}$ and a theory $\mathcal{T}$ over the language $\mathcal{L}$ by adding symbols for arbitrary functions f (or relations R) to $\mathcal{L}$ and their defining axioms A_f (or A_R) to the theory $\mathcal{T}$. If the appropriate functions are definable in $\mathcal{T}$ (according to Definition A.20) then the theory $\mathcal{T} + A_f$ $(+A_R)$ is conservative over $\mathcal{T}$. This enables one to add new function and relation symbols to the language while proving statement inside a theory; as long as these function and relation symbols are definable in the theory, every statement in the original language proved in the extended theory (with the additional defining-axioms for the functions and relations) is provable in the original theory over the original language. *However*, extending the language and the theory in such a way *does not guarantee* that one can use the new function symbols in the comprehension (and induction) axiom schemes. In other words, using the comprehension (and induction) axioms over the expanded language might not result in a conservative extension. Therefore, definability will not be enough for our purposes. We will show precisely in the sequel (Sections A.2.1 and A.2.2) how to make sure that a function is both definable in the theories we work with and also can be used in the corresponding comprehension and induction axiom schemes (while preserving conservativity).

When expanding the language with new function symbols we can assume that in *bounded formulas* the bounding terms possibly use function symbols from the the expanded language.[3]

Extending $\mathbf{V}^0$ with new function and relation symbols

Here we describe a process (presented in Section V.4. in [29]) by which we can extend the language $\mathcal{L}_A^2$ of $\mathbf{V}^0$ by new function symbols, obtaining a conservative extension of $\mathbf{V}^0$ that can also prove the comprehension and induction axiom schemes in the extended language.

First note that every relation or function symbol has an intended or standard interpretation over the standard model $\mathbb{N}_2$ (for instance, the standard interpretation of the binary function "+" is that of the addition of two natural numbers). If not explicitly defined otherwise, we will always assume that a defining axiom of a symbol in the language defines a symbol in a way that its interpretation in $\mathbb{N}_2$ is the standard one. Note also that we shall use the same symbol $F(\vec{x}, \vec{X})$ to denote a function and the function *symbol* in the (extended) language in the theory.

Definition A.22 (Relation representable in a language)**.** Let Φ be a set of formulas in a language $\mathcal{L}$ extending $\mathcal{L}_A^2$. We say a relation $R(\vec{x}, \vec{X})$ is *representable* by a formula from Φ iff there is a formula $\varphi(\vec{x}, \vec{X}, Y)$ in Φ such that in the standard two-sorted model $\mathbb{N}_2$ (and when all relation and function symbols in $\mathcal{L}$ get their intended interpretation), it holds that:

$$R(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}). \tag{A.8}$$

We say that a number function $f(\vec{x}, \vec{X})$ is *polynomially-bounded* if $f(\vec{x}, \vec{X}) \leq \mathrm{poly}(\vec{x}, |\vec{X}|)$. We say that a string function $F(\vec{x}, \vec{X})$ is *polynomially-bounded* if $|F(\vec{x}, \vec{X})| \leq \mathrm{poly}(\vec{x}, |\vec{X}|)$.

[3]Because any definable function in a bounded theory can be bounded by a term in the original language $\mathcal{L}_A^2$ (cf. [29]).

Definition A.23 (Bit-definition)**.** Let $F(\vec{x}, \vec{X})$ be a polynomially-bounded string function. We define the *bit-graph of* F to be the relation $R(i, \vec{x}, \vec{X})$, where i is a number variable, such that

$$F(\vec{x}, \vec{X})(i) \leftrightarrow i < t(\vec{x}, \vec{X}) \wedge R(i, \vec{x}, \vec{X}),$$

for some number term $t(\vec{x}, \vec{X})$.

Definition A.24 (Σ_0^B-definability from a language; Definition V.4.1.2. in [29])**.** We say that a number function f is Σ_0^B-definable from a language $\mathcal{L} \supseteq \mathcal{L}_A^2$, if f is polynomially-bounded and its graph is represented by a $\Sigma_0^B(\mathcal{L})$ formula φ. We call the formula φ the *defining axiom of* f. We say that a string function F is Σ_0^B-definable from a language $\mathcal{L} \supseteq \mathcal{L}_A^2$, if F is polynomially-bounded and its bit-graph is representable by a $\Sigma_0^B(\mathcal{L})$ formula φ. We call the formula φ the *defining axiom of* F or the *bit-defining axiom of* F.

Note: We used the term *defining axiom of a function* f in both the case where f is defined *from a language* (Definition A.24) and in case f is definable *in the theory* (Definition A.20). We will show in the sequel that for our purposes these two notions coincide: when we define a function from a language the function will be definable also in the relevant theory, and so the defining axiom of f from the language will be the defining axiom of f in the theory (when the theory is possibly extended conservatively to include new function symbols).

Also, note that if the graph of a function F is representable by a $\Sigma_0^B(\mathcal{L})$ formula then clearly also the bit-graph of F is representable by a $\Sigma_0^B(\mathcal{L})$ formula. Therefore, *it suffices to show a* $\Sigma_0^B(\mathcal{L})$ *formula representing the graph of a function* F *to establish that* F *is* Σ_0^B*-definable from* $\mathcal{L}$.

Definition A.25 ($\mathbf{AC}^0$-reduction)**.** A number function f is $\mathbf{AC}^0$*-reducible to* $\mathcal{L} \supseteq \mathcal{L}_A^2$ iff there is a possibly empty sequence of functions $F_1, \ldots, F_k$ such that F_i is Σ_0^B-definable from $\mathcal{L} \cup \{F_1, \ldots, F_{i-1}\}$, for any $i = 1, \ldots, k$, and f is Σ_0^B-definable from $\mathcal{L} \cup \{F_1, \ldots, F_k\}$.

We now describe the standard process enabling one to extend a theory $\mathcal{T} \supseteq \mathbf{V}^0$ over the language $\mathcal{L}_A^2$ with new function symbols obtaining a conservative extension of $\mathcal{T}$ such that the new function symbols can also be used in comprehension and induction axiom schemes in the theory (see Section V.4. in [29] for the proofs):

(i) If the number function f is Σ_0^B-definable from $\mathcal{L}_A^2$, then $\mathcal{T}$ over the language $\mathcal{L}_A^2 \cup \{f\}$, augmented with the defining axiom of f, is a conservative extension of $\mathcal{T}$ and we can also prove the comprehension and induction axioms for $\Sigma_0^B(f)$ formulas.

(ii) If the string function F is Σ_0^B-definable from $\mathcal{L}_A^2$, then $\mathcal{T}$ over the language $\mathcal{L}_A^2 \cup \{F\}$, augmented with the bit-defining axiom of F, is a conservative extension of $\mathcal{T}$ and we can also prove the comprehension and induction axioms for $\Sigma_0^B(F)$ formulas.

(iii) We can now iterate the above process of extending the language $\mathcal{L}_A^2(f)$ (or equivalently, $\mathcal{L}_A^2(F)$) to conservatively add more functions $f_2, f_3, \ldots$ to the language, which can also be used in comprehension and induction axioms.

By the aforementioned and by Definition A.25, we can extend the language of a theory with a new function symbol f, *whenever* f *is* $\mathbf{AC}^0$*-reducible to* $\mathcal{L}_A^2$. This results in an extended theory (in an extended language) which is conservative, and can prove the comprehension and induction axioms for formulas in the extended language. In the sequel, when defining a new function in $\mathbf{V}^0$ we may simply say that it is Σ_0^B*-definable (or bit-definable) in* $\mathbf{V}^0$ and give its Σ_0^B-defining (bit-defining, respectively) axiom (that can possibly use also previously Σ_0^B-defined (or bit defined) function symbols).

Extending the language of $\mathbf{V}^0$ with new *relation* symbols is simple: every relation $R(\vec{x}, \vec{X})$ which is representable by a $\Sigma_0^B(\mathcal{L})$ formula, where $\mathcal{L}$ is an extension of the language with new function symbols obtained as shown above, can be added itself to the language. This results in a conservative extension of $\mathbf{V}^0$ that also proves the Σ_0^B induction and comprehension axioms in the extended language.

Definition A.26 ($\mathbf{FAC}^0$)**.** A string (number) function is in $\mathbf{FAC}^0$ if it is polynomially bounded and its bit-graph (graph, respectively) is definable by a Σ_0^B formula in the language $\mathcal{L}_A^2$.

Basic formalizations in $\mathbf{V}^0$

In this section we show how to formalize basic notions in the theory $\mathbf{V}^0$.

Characteristic function of a relation. For a given predicate R we denote by χ_R the *characteristic function* of R. If R is Σ_0^B-definable in $\mathbf{V}^0$ then χ_R is Σ_0^B-definable in $\mathbf{V}^0$, using the following defining axiom:

$$y = \chi_R(\vec{x}, \vec{X}) \leftrightarrow \Big(R(\vec{x}, \vec{X}) \rightarrow y = 1 \wedge \neg R(\vec{x}, \vec{X}) \rightarrow y = 0\Big).$$

Natural number sequences of constant length. For two numbers x, y let $\langle x, y\rangle := (x + y)(x + y + 1) + 2y$ be the *pairing function*, and let *left*(z), *right*(z) be the (easily Σ_0^B-definable in $\mathbf{V}^0$) projection functions of the first and second element in the pair z, respectively. It should be clear from the context when we mean $\langle a, b\rangle$ as an inner product of two vectors and when we mean it as the pairing function. We also Σ_0^B-define inductively $\langle v_1, \ldots, v_k\rangle := \langle\langle v_1, \ldots, v_{k-1}\rangle, v_k\rangle$, for any constant k. Then $\mathbf{V}^0$ proves the injectivity of the pairing function and lets us handle such pairs in a standard way.

Notation: Given a number x, coding a sequence of natural numbers of length k, we write $\langle x\rangle_i^k$, for $i = 1, \ldots, k$, to denote the number in the ith position in x. This is a Σ_0^B-definable function in $\mathbf{V}^0$ (defined via *left*(x), *right*(x) functions).

Rational numbers. Given the natural numbers, we can define the integers in $\mathbf{V}^0$ by identifying an integer number with a pair $\langle a, b\rangle$, such that a is its "positive" part and b is its "negative" part. We can define addition, product and subtraction of integers. All with Σ_0^B definitions.

Having the integer numbers, we define the rational numbers as follows: for two integer numbers a, b, the rational number a/b, is defined by the pair $\langle a, b\rangle$. We can define addition, subtraction and multiplication of rational numbers in $\mathbf{V}^0$ by

Σ_0^B definitions. (See for example in [69]). However, we shall take a simpler path in this paper: ***throughout this paper, all rational numbers used inside the theories have the same denominator*** n^{2c}***, for some fixed constant*** c. This enables us to represent every rational number with a pair of integer numbers, such that each has a value polynomial in n. Addition and multiplication of two rational numbers is also Σ_0^B-definable in $\mathbf{V}^0$. This also makes it more convenient to sum a non-constant number of rational numbers in $\mathbf{VTC}^0$ (see Proposition A.33). To keep the invariant that all denominators are n^{2c}, we then make sure that all the rational numbers resulting from computation in the proof in the theory are indeed integer products of $1/n^{2c}$. This will hold since by inspection of the computations made in the theory it will be clear that:

1. all initial rational numbers will be integer products of $1/n^c$;
2. all arithmetic operations done on rational numbers are one of the following:
   (a) addition of two rational numbers (this preserves the denominator);
   (b) if we multiply two rational numbers x, y then $x = \frac{n^c \cdot a}{n^{2c}}$ and $y = \frac{n^c \cdot b}{n^{2c}}$ for some two integers a, b, and so $x \cdot y = \frac{ab}{n^{2c}}$ will have n^{2c} as a denominator.

Convention: For the sake of readability we sometimes treat an integer number m in the theory as its corresponding rational number $m/1$, thus enabling one to compute with both types. (This is easy to achieve formally. E.g., one can define a function $numones'(X)$ that outputs the corresponding rational number of the integer $numones(X)$.)

Absolute numbers. We can Σ_0^B-define in $\mathbf{V}^0$ the absolute value function for integer numbers $abs_{\mathbb{Z}}(\cdot)$ from the language $\mathcal{L}_A^2$ as follows (the function max is easily Σ_0^B-definable):

$$y = abs_{\mathbb{Z}}(x) \leftrightarrow y = \langle max(left(x) - right(x), right(x) - left(x)), 0\rangle.$$

We Σ_0^B-define the absolute value function for rational numbers $abs_{\mathbb{Q}}(\cdot)$ in $\mathbf{V}^0$ as follows:

$$y = abs_{\mathbb{Q}}(x) \leftrightarrow y = \langle abs_{\mathbb{Z}}(left(x)), \langle n^{2c}, 0\rangle\rangle.$$

For simplicity, we shall suppress the subscript $\mathbb{Z}, \mathbb{Q}$ in $abs_{\mathbb{Z}}, abs_{\mathbb{Q}}$; the choice of function can be determined from the context.

Number (natural, integers and rational) sequences of polynomial length. If we wish to talk about sequences of numbers (whether natural, integers or rationals) where the lengths of the sequences are non-constant, we have to use string variables. Using the number tupling function we can encode sequences as sets of numbers (recall that a string is identified with the finite set of numbers encoding it). Essentially, a sequence is encoded as a string Z such that the xth number in the sequence is y if the number $\langle x, y\rangle$ is in Z. Formally we have the following Σ_0^B-defining formula for the function $seq(x, Z)$:

$$\begin{aligned} y = seq(x, Z) \leftrightarrow (y < |Z| \wedge Z(\langle x, y\rangle) \wedge \forall z < y \neg Z(\langle x, z\rangle)) \\ \vee (\forall z < |Z| \neg Z(\langle x, z\rangle) \wedge y = |Z|). \end{aligned} \tag{A.9}$$

Formula (A.9) states that the xth element in the sequence coded by Z is y iff $\langle x, y\rangle$ is in Z *and* no other number smaller than y also "occupies the xth position in the sequence", and that if no number occupies position x then the function returns the length of the string variable Z. We write $Z[x]$ to abbreviate $seq(x, Z)$.

According to the definition of the function $seq(x, Z)$ above, there might be more than one string Z that encodes the same sequence of numbers. However, we sometimes need to determine a *unique* string encoding a sequence. To this end we use a Σ_0^B formula, denoted $SEQ(y, Z)$, which asserts that Z is the lexicographically smallest string that encodes a sequence of $y + 1$ numbers (i.e., no string with smaller binary code encodes the same sequence). Specifically, the formula states that if $w = \langle i, j\rangle$ is in Z then j is indeed the ith element in the sequence coded by Z, and for all $y \geq j$ the pair $\langle i, y\rangle$ is not contained in Z:

$$SEQ(y, Z) \equiv \forall w < |Z|\,(Z(w) \leftrightarrow \exists i \leq y \exists j < |Z|\,(w = \langle i, j\rangle \wedge j = Z[i]))\,. \tag{A.10}$$

Note that elements of sequences Z coded by strings are referred to as $Z[i]$, while elements of sequences x coded by a number are referred to as $\langle x\rangle_i^k$ (for k the length of the sequence x). We define the number function $length(Z)$ to be the length of the sequence Z, as follows:

$$\ell = length(Z) \leftrightarrow SEQ(\ell, Z) \wedge \exists w < |Z| \exists j < |Z|(Z(w) \wedge w = \langle \ell - 1, j\rangle)\,.$$

The defining axiom of $length(Z)$ states that Z encodes a sequence and is the lexicographically smallest string that encodes this sequence and that the largest position in the sequence which is occupied is $\ell - 1$ (by definition there will be no pair $\langle a, b\rangle \in Z$ with $a > \ell - 1$).

Array of strings. We want to encode a sequence of strings as an array. We use the relation $RowArray(x, Z)$ to denote the xth string in Z as follows (we follow the treatment in [29], Definition V.4.26, page 114).

Definition A.27 (Array of strings)**.** The function $RowArray(x, Z)$, denoted $Z^{[x]}$, is Σ_0^B-definable in $\mathbf{V}^0$ using the following bit-definition:[4]

$$RowArray(x, Z)(i) \;\leftrightarrow\; (i < |Z| \wedge Z(\langle x, i\rangle)).$$

We will abuse notation and write $length(Z)$ for the length of the array Z (i.e., numbers of strings in Z) even when Z is a $RowArray$ (and not a sequence according to the predicate SEQ).

Functions for constructing sequences.

Definition A.28 ($Sequence_f(y, \vec{x}, \vec{X})$)**.** Let $f(z, \vec{x}, \vec{X})$ be a Σ_0^B-definable number function in $\mathbf{V}^0$ (or a Σ_1^B-definable number function in $\mathbf{VTC}^0$[see section A.2.2 below]), then $Sequence_f(y, \vec{x}, \vec{X})$ is the string function Σ_0^B-definable in $\mathbf{V}^0$(or Σ_1^B-definable in $\mathbf{VTC}^0$, respectively) that returns the number sequence whose jth position is $f(j, \vec{x}, \vec{X})$, for $j = 0, \ldots, y$.

[4]We use the name "RowArray" (instead of the name "Row" used in [29]).

In other words, $Sequence_f(y, \vec{x}, \vec{X})$ returns the graph of the function $f(z, \vec{x}, \vec{X})$ up to y (that is, the sequence $\langle f(0, \vec{x}, \vec{X}), \ldots, f(y, \vec{x}, \vec{X})\rangle$). The following is the Σ^B_0-definition of the $Sequence_f(y, \vec{x}, \vec{X})$:

$$Y = Sequence_f(y, \vec{x}, \vec{X}) \leftrightarrow SEQ(y, Y) \wedge \forall z \leq y\,(Y[i] = f(z, \vec{x}, \vec{X})).$$

Sequences of numbers with higher-dimensions. For a constant k, let S be a k-dimensional sequence of rational numbers. We encode a sequence S as a string variable Z such that the $\langle i_1, \ldots, i_k\rangle$th element in S is extracted by the function seq (defined above). Specifically, we have $S[\langle i_1, \ldots, i_k\rangle] = y$ iff $\langle\langle i_1, \ldots, i_k\rangle, y\rangle \in Z$ and there is no $z < y$ for which $\langle\langle i_1, \ldots, i_k\rangle, z\rangle \in Z$. Accordingly, we write $Z[i_1, \ldots, i_k]$ to abbreviate $seq(\langle i_1, \ldots, i_k\rangle, Z)$.

Matrices. Given a rational $n \times n$ matrix M, we define it as a two-dimensional sequence in the manner defined above; and refer to the number at row $1 \leq i \leq n$ and column $1 \leq j \leq n$ of M as $M[i, j]$. We can define the *string* function that extracts the xth row of M, and the xth column of M, respectively, with Σ^B_0 formulas as follows. First define $f(M, i, x) := M[i, x]$, $g(M, i, x) := M[x, i]$, for any $i = 0, 1, \ldots, n$ (for $i = 0$, the value of $M[i, x]$ and $M[x, i]$ does not matter; but this value is still defined by definition of the function seq). Then use Definition A.28 to define:

$$\begin{aligned} Row(i, M) &:= Sequence_f(i, n) \\ Column(i, M) &:= Sequence_g(i, n)\,. \end{aligned}$$

A.2.2 The theory $\mathbf{VTC}^0$

It is known that $\mathbf{V}^0$ is incapable of proving basic counting statements. Specifically, it is known that the function that sums a sequence of numbers (of non-constant length) is not provably total, namely, is not Σ^B_1-definable in $\mathbf{V}^0$. Therefore, if a proof involves such computations we might not be able to perform it in $\mathbf{V}^0$. The theory $\mathbf{VTC}^0$ extends $\mathbf{V}^0$, and is meant to allow reasoning that involves counting, and specifically to sum a non-constant sequence of numbers. The theory $\mathbf{VTC}^0$ was introduced in [70]; we refer the reader to Section IX.3.2 [29] for a full treatment of this theory. The Σ^B_0 theorems of $\mathbf{VTC}^0$ correspond to polynomial-size TC^0-Frege propositional proofs, which will enable us to prove the main result of this paper.

Definition A.29 (NUMONES)**.** Let $\delta_{\mathsf{NUM}}(y, X, Z)$ be the following Σ^B_0 formula:

$$\begin{aligned} \delta_{\mathsf{NUM}}(y, X, Z) := SEQ(y, Z) \wedge Z[0] = 0 \wedge \forall u < y((X(u) \rightarrow Z[u+1] = Z[u] + 1) \\ \wedge (\neg X(u) \rightarrow Z[u+1] = Z[u])). \end{aligned} \tag{A.11}$$

Define NUMONES to be the following Σ^B_1 formula:

$$\mathsf{NUMONES} := \exists Z \leq 1 + \langle y, y\rangle \delta_{\mathsf{NUM}}(y, X, Z). \tag{A.12}$$

Informally one can think of the sequence $Z(X)$, which existence is guaranteed by $\mathsf{NUMONES}$, as a sequence counting the number of ones in a string X, that is, the uth entry in $Z(X)$ is the number of ones appearing in the string X up to the uth position.

Definition A.30 ($\mathbf{VTC}^0$)**.** The theory $\mathbf{VTC}^0$ is the theory containing all axioms of $\mathbf{V}^0$ and the axiom $\mathsf{NUMONES}$.

Using $\mathsf{NUMONES}$ we can define the function $numones(y, X)$ that, given y and X, returns the yth entry of $Z(X)$ via the following Σ^B_1-defining axiom

$$numones(y, X) = z \leftrightarrow \exists Z \leq 1 + \langle |X|, |X| \rangle \left(\delta_{\mathsf{NUM}}(|X|, X, Z) \wedge Z[y] = z\right). \tag{A.13}$$

We shall use the following abbreviation:

$$numones(X) := numones(|X| - 1, X).$$

Next we show how to obtain the functions we will use in the theory $\mathbf{VTC}^0$ (these will include the function $numones$).

Extending $\mathbf{VTC}^0$ with new function and relation symbols

Similar to the case of $\mathbf{V}^0$, we would like to extend the language $\mathcal{L}^2_A$ of $\mathbf{VTC}^0$ with new function and relation symbols, to obtain a conservative extension. Moreover, we require that the new function and relation symbols could be used in induction and comprehension axioms (while preserving conservativity). We can do this, using results from Sections I.X.3.2 and I.X.3.3 in [29], as follows.

Definition A.31 (Number summation)**.** For any number function $f(z, \vec{x}, \vec{X})$ define the number function $\mathsf{sum}_f(y, \vec{x}, \vec{X})$ by[5]

$$\mathsf{sum}_f(y, \vec{x}, \vec{X}) = \sum_{i=0}^{y} f(i, \vec{x}, \vec{X}).$$

Recall that by Definition A.24, a string (number) function F is Σ^B_0-*definable from* $\mathcal{L} \supseteq \mathcal{L}^2_A$ iff there is a Σ^B_0 formula over the language $\mathcal{L}$ that bit-defines (defines, respectively) the function F (when all the functions and relation symbols in $\mathcal{L}$ get their intended interpretation).

We can use the following facts to extend the language of $\mathbf{VTC}^0$ with new function symbols (proved in Section IX.3.2 in [29]): if f is a (number or string) function in $\mathbf{FTC}^0$ (see below), then there is a Σ^B_1 formula φ that represents its graph, and the theory $\mathbf{VTC}^0$ extended with the defining axiom for f (using φ, as in Definition A.24) over the language $\mathcal{L} = \mathcal{L}^2_A \cup \{f\}$ is a conservative extension of $\mathbf{VTC}^0$. And by Theorem IX.3.7 in Section IX.3.2 [29], $\mathbf{VTC}^0$ can prove the induction and comprehension axioms for any $\Sigma^B_0(\mathcal{L})$ formula.

Thus, to extend $\mathbf{VTC}^0$ with new *function* symbols, by the above it suffices to show how to obtain $\mathbf{FTC}^0$ functions. For this we use the following equivalent characterizations of $\mathbf{FTC}^0$ (see Sections IX.3.2 and IX.3.3 in [29]):

[5]Note that this is a definition in the metatheory (or in other words the standard two-sorted model).

Proposition A.32 (Theorem IX.3.12, Proposition IX.3.1 in [29])**.** The following statements are equivalent:

1. The function f is Σ_1^B-definable in $\mathbf{VTC}^0$, and is applicable inside comprehension and induction axiom schemes.
2. The function f is in $\mathbf{FTC}^0$.
3. The function f is obtained from $\mathbf{FAC}^0$ by number summation and $\mathbf{AC}^0$-reductions.
4. There exist a natural k and functions $f_1, \ldots, f_k = f$ such that for every $i = 1, \ldots, k$, the function f_i is either definbale by a Σ_0^B formula in the language $\mathcal{L}_A^2 \cup \{f_1, \ldots, f_{i-1}\}$ or there exists $h \in \mathcal{L}_A^2 \cup \{f_1, \ldots, f_{i-1}\}$ such that $f_i = \mathsf{sum}_h$.
5. The function f is $\mathbf{AC}^0$-reducible to $\mathcal{L}_A^2 \cup \{numones\}$.

Therefore, to obtain new $\mathbf{FTC}^0$ functions, and hence to extend conservatively the language of $\mathbf{VTC}^0$ with function symbols that can also be used in comprehension and induction axioms, we can define a function with a Σ_0^B formula in a language that contains sum_f, for f in $\mathbf{FAC}^0$, and possibly contains also other symbols already definable in $\mathbf{V}^0$. Then, we can iterate this process a finite number of times, where now sum_f is defined also for f being a function defined in a previous iteration. Since a function is in $\mathbf{FTC}^0$ iff it is Σ_1^B-definable in $\mathbf{VTC}^0$, new functions obtained in this way, are said to be Σ_1^B*-definable in* $\mathbf{VTC}^0$.

To extend the language of $\mathbf{VTC}^0$ with new *relation* symbols, we can simply add new Σ_0^B-definable relations, using possibly relation and function symbols that where already added before to the language, and specifically the *numones* function. Such relations can then be used in induction and comprehension axioms, and we shall say that they are Σ_0^B*-definable relations in* $\mathbf{VTC}^0$.

Summation in $\mathbf{VTC}^0$

Here we show how to express and prove basic equalities and inequalities in the theory $\mathbf{VTC}^0$.

Summation over natural and rational number sequences. Given a sequence X of *natural* numbers, we define the function that sums the numbers in X until the yth position by $\mathsf{sum}_{seq}(y, X)$ which is equal to $\sum_{i=0}^{y} seq(i, X)$.

To sum sequences of *rational* numbers, on the other hand, we do the following. For our purposes it is sufficient to sum many small (that is, polynomially bounded) numbers (this is in contrast to additions of numbers encoded as *strings*). Recall that we assume that all rational numbers in the theory have the same denominator n^{2c}, for some global constant c, independent of n.

Proposition A.33. Let X be a sequence of rational numbers with denominator n^{2c} and let $sum_{\mathbb{Q}}(z, X)$ be the number function that outputs $\sum_{i=0}^{z} X[i]$. Then, the number function $sum_{\mathbb{Q}}(z, X)$ is Σ_1^B-definable in $\mathbf{VTC}^0$.

Proof. It suffices to show that there is a Σ_0^B formula that defines the number function $sum_{\mathbb{Q}}(z, X)$ using only number summation functions and $\mathbf{FAC}^0$ functions.

The $\mathbf{AC}^0$ function $seq(i, X)$ extracts the ith element (that is, rational number) from the sequence X (see Formula (A.9)). A rational number is a pair of integers, and hence is a pair of pairs. Thus, $gp(i, X) := left(left(seq(i, X)))$ extracts the positive part of the integer numerator of the ith rational number in X, and $gn(i, X) := right(left(seq(i, X)))$ extracts the negative part of the integer numerator of the ith rational number in X. Note that both $gp(i, X)$ and $gn(i, X)$ are $\mathbf{FAC}^0$ functions. Therefore, $\mathsf{sum}_{gp}(z, X)$ equals the sum of all the positive parts in X, and $\mathsf{sum}_{gn}(z, X)$ equals the function that sums of all the negative parts of the numerators in X. We can now define $sum_{\mathbb{Q}}(z, X)$ as follows:

$$w = sum_{\mathbb{Q}}(z, X) \leftrightarrow w = \left\langle \left\langle \mathsf{sum}_{gp}(z, X), \mathsf{sum}_{gn}(z, X) \right\rangle, \left\langle n^{2c}, 0 \right\rangle \right\rangle \tag{A.14}$$

Note indeed that $\langle \langle \mathsf{sum}_{gp}(z, X), \mathsf{sum}_{gn}(z, X) \rangle, \langle n^{2c}, 0 \rangle \rangle$ is a pair of integers that encodes the desired rational number (with denominator n^{2c}). □

Notation: As a corollary from Proposition A.33, we can abuse notation as follows: for $f(y, \vec{x}, \vec{X})$ a number function mapping to the rationals we write $\mathsf{sum}_f(n, \vec{x}, \vec{X})$ to denote the sum of *rationals* $\sum_{i=0}^{n} f(i, \vec{x}, \vec{X})$, for some fixed $\vec{x}, \vec{X}$ and n. Abusing notation further, we can write in a formula in the theory simply $\sum_{i=0}^{n} f(i, \vec{x}, \vec{X})$.

Expressing vectors and operations on vectors. Vectors over $\mathbb{Q}$ are defined as sequences of rational numbers (for simplicity we shall assume that the number at the 0 position of a vector is 0). Given two rational vectors $\mathbf{v}, \mathbf{u}$ of size n, their inner prduct, denoted $\langle \mathbf{v}, \mathbf{u} \rangle$, is defined as follows (we identify here $\mathbf{v}, \mathbf{u}$ with the string variables encoding $\mathbf{v}, \mathbf{u}$): let $f(y, \mathbf{v}, \mathbf{u})$ be the $\mathbf{FAC}^0$ number function defined by $f(y) := \mathbf{v}[y] \cdot \mathbf{u}[y]$. Then the inner product of $\mathbf{v}$ and $\mathbf{u}$ is defined by

$$innerprod(\mathbf{v}, \mathbf{u}) := sum_{\mathbb{Q}} \left(length(\mathbf{v}) + 1, Sequence_f(length(\mathbf{v}) + 1) \right).$$

The function that adds two rational vectors is easily seen to be in $\mathbf{FAC}^0$ (use Definition A.28 to construct a sequence, where each entry in the sequence is the addition of the corresponding entries of the two vectors).

Expressing product of matrices and vectors. Let $\mathbf{v}$ be an n-dimensional rational vector and let M be an $n \times n$ rational matrix. Assume that $f(z, M, \mathbf{v}) := innerprod(Row(z, M), \mathbf{v})$. We Σ_1^B-define in $\mathbf{VTC}^0$ the product $M\mathbf{v}$ as follows:

$$Matvecprod(M, \mathbf{v}) := Sequence_f(length(\mathbf{v}) + 1, M, \mathbf{v}).$$

Notation: When reasoning in the theory $\mathbf{VTC}^0$ we sometimes abuse notation and write $\mathbf{v} \cdot \mathbf{u}$ or $\langle \mathbf{v}, \mathbf{u} \rangle$ instead of $innerprod(\mathbf{u}, \mathbf{v})$, and $M\mathbf{v}$ instead of $Matvecprod(M, \mathbf{v})$, and $\mathbf{u}^t M\mathbf{v}$ instead of $\langle \mathbf{u}, M\mathbf{v} \rangle$.

Counting in $\mathbf{VTC}^0$

Here we present basic statements involving counting of certain objects and sets, provable in $\mathbf{VTC}^0$.

Notation: When reasoning in the theory $\mathbf{VTC}^0$, we will say that a family of Σ_0^B-definable in $\mathbf{VTC}^0$ sets $B_0, \ldots, B_\ell$ *forms a partition of a set* B whenever $\mathbf{VTC}^0$ proves that (i) $\bigcup_{i=0}^{\ell} B_i = B$, and (ii) $B_i \cap B_j = \emptyset$, for all $0 \leq i \neq j \leq \ell$. Here, $\bigcup_{i=0}^{\ell} B_i := \{r \ : \ \exists i \leq \ell, B_i(r)\}$.

Proposition A.34 (Some counting in $\mathbf{VTC}^0$)**.** Let $B_1, \ldots, B_\ell$ be family of Σ_0^B-definable sets in $\mathbf{VTC}^0$ that partition the set B (ℓ may be a variable). Then, $\mathbf{VTC}^0$ proves:

$$numones(B) = \sum_{i=1}^{\ell} numones(B_i)\,.$$

Proof. We proceed by induction on ℓ to show that
for every $0 \leq y \leq \max\{B_1, \ldots, B_\ell\}$:

$$numones(y, B_1 \cup \ldots \cup B_\ell) = \sum_{i=1}^{\ell} numones(y, B_i).$$

Base case: $\ell = 1$. Thus, $B = B_1$ and so we need to prove only $numones(y, B_1) = \sum_{i=1} numones(y, B_i)$. Since $\mathbf{VTC}^0$ proves that a summation that contains only one summand B_1 equals B_1 we are done.

Induction step: $\ell > 1$. We have $B = \bigcup_{i=1}^{\ell} B_i = (\bigcup_{i=1}^{\ell-1} B_i) \cup B_\ell$. Assume by way of contradiction that $(\bigcup_{i=1}^{\ell-1} B_i) \cap B_\ell \neq \emptyset$. Then $\mathbf{VTC}^0$ can prove that this contradicts the assumption that $B_i \cap B_j = \emptyset$, for all $i \neq j$ (which holds since the B_i's form a partition of B). Hence, $(\bigcup_{i=1}^{\ell-1} B_i) \cap B_\ell = \emptyset$, and by Claim A.35 (proved below):

$$numones(y, B) = numones(y, \bigcup_{i=1}^{\ell-1} B_i) + numones(y, B_\ell) \tag{A.15}$$

$$= \sum_{i=1}^{\ell-1} numones(y, B_i) + numones(y, B_\ell) \quad \text{(by induction hypothesis)} \tag{A.16}$$

$$= \sum_{i=1}^{\ell} numones(y, B_i). \tag{A.17}$$

It remains to prove the following:

Claim A.35. (In $\mathbf{VTC}^0$) let A, B be two sets such that $A \cap B = \emptyset$, then for all $0 \leq y \leq \max\{|A|, |B|\}$:

$$numones(y, A \cup B) = numones(y, A) + numones(y, B).$$

Proof of claim: We proceed by induction on y, using the defining axiom of *numones* (stating the existence of a counting sequence for the input string variable; see Equations (A.13) and (A.11)).

Base case: $y = 0$. The counting sequence Z for $numones(A \cup B)$ is defined such that $Z[0] = 0$. Thus,

$$0 = numones(0, A \cup B) = numones(0, A) + numones(0, B) = 0 + 0 = 0.$$

Induction step: $0 < y \leq \max\{|A|, |B|\}$. By the defining axiom of *numones* we have:

$$numones(y, A \cup B) = \begin{cases} numones(y-1, A \cup B) + 1, & y \in A \cup B; \\ numones(y-1, A \cup B), & \text{otherwise.} \end{cases} \tag{A.18}$$

We have to consider the following three cases:
Case 1: $y \in A$. Thus, by assumption that A and B are disjoint, we have $y \notin B$. Also, we have $y \in A \cup B$. Therefore:

$$\begin{aligned} &numones(y, A) + numones(y, B) \\ &\quad = numones(y-1, A) + 1 + numones(y, B) && \text{(since } y \in A\text{)} \\ &\quad = numones(y-1, A) + 1 + numones(y-1, B) && \text{(since } y \notin B\text{)} \\ &\quad = numones(y-1, A \cup B) + 1 && \text{(by induction hypothesis)} \\ &\quad = numones(y, A \cup B) && \text{(since } y \in A \cup B\text{).} \end{aligned}$$

Case 2: $y \in B$. This is the same as Case 1.
Case 3: $y \notin A \cup B$. This is similar to the previous cases. We omit the details.
■Claim

□

Proposition A.36 (More counting in $\mathbf{VTC}^0$)**.** Let $\varphi(x)$ be a Σ_0^B formula (possibly in an extended language of $\mathbf{VTC}^0$). The theory $\mathbf{VTC}^0$ can prove that if $Z = \{0 \leq i < m : \varphi(i)\}$ and for any $0 \leq i < m$,

$$\gamma_i = \begin{cases} a, & \varphi(i); \\ b, & \neg\varphi(i), \end{cases}$$

then

$$\sum_{i<m} \gamma_i = a \cdot numones(Z) + b \cdot (m - numones(Z)).$$

Proof. Since $\varphi(x)$ is a Σ_0^B formula, by Section A.2.2, we can use the comprehension axiom scheme to define, for any $0 \leq k \leq m-1$, the set:

$$Z_k := \{i \leq k : \varphi(i)\}.$$

The claim is proved by induction on k.

Base case: $k = 0$. If $\varphi(0)$ is true, then $Z_0 = \{0\}$, and so $numones(Z_0) = 1$. By assumption we have $\gamma_0 = a = a \cdot numones(Z_0) + b \cdot (1 - numones(Z_0))$.

Otherwise, $\varphi(0)$ is false and so $Z_0 = \emptyset$, implying that $numones(Z_k) = 0$. By assumption again we have $\gamma_0 = b = a \cdot numones(Z_0) + b(1 - numones(Z_0))$.

Induction step: $k > 0$.
Case 1: $\varphi(k)$ is true. Thus $Z_k(k)$ is true and also

$$numones(Z_k) = numones(Z_{k-1}) + 1, \tag{A.19}$$

and by assumption $\gamma_k = a$. Therefore,

$$\begin{aligned}
&\sum_{i=0}^{k} \gamma_i = \sum_{i=0}^{k-1} \gamma_i + \gamma_k = \sum_{i=0}^{k-1} \gamma_i + a \\
&= a \cdot numones(Z_{k-1}) + b \cdot (k - 1 - numones(Z_{k-1})) + a && \text{(by induction hypothesis)} \\
&= a \cdot (numones(Z_{k-1}) + 1) + b \cdot (k - 1 - numones(Z_{k-1})) && \text{(rearranging)} \\
&= a \cdot numones(Z_k) + b \cdot (k - numones(Z_k)) && \text{(by (A.19)).}
\end{aligned}$$

Case 2: $\varphi(k)$ is false. This is similar to Case 1. Specifically, $Z_k(k)$ is false and also

$$numones(Z_k) = numones(Z_{k-1}), \tag{A.20}$$

and by assumption $\gamma_k = b$. Therefore

$$\begin{aligned}
&\sum_{i=0}^{k} \gamma_i = \sum_{i=0}^{k-1} \gamma_i + \gamma_k = \sum_{i=0}^{k-1} \gamma_i + b \\
&= a \cdot numones(Z_{k-1}) + b \cdot (k - 1 - numones(Z_{k-1})) + b && \text{(by induction hypothesis)} \\
&= a \cdot numones(Z_{k-1}) + b \cdot (k - 1 - numones(Z_{k-1}) + 1) && \text{(rearranging)} \\
&= a \cdot numones(Z_k) + b \cdot (k - numones(Z_k)) && \text{(by (A.20)).}
\end{aligned}$$

□

For a number term t, we write $\forall x \in [t]\, \Phi$ to abbreviate $\forall x \leq t(x \geq 1 \to \Phi)$. We shall use the following proposition in Section A.4 (Lemma A.56).

Proposition A.37. The theory $\mathbf{VTC}^0$ proves the following statement. Let $F(x)$ be a string function. Let $d < t$ be a natural number and assume that any number in any set $F(1), \ldots, F(t)$ occurs in at most d many sets in $F(1), \ldots, F(t)$. Let $g(x)$ be a number function such that $g(1), \ldots, g(t)$ are (not necessarily distinct) numbers with $g(i) \in F(i)$ for all $i \in [t]$. Then $numones(\{g(i) \,:\, i \in [t]\}) \geq \lceil t/d \rceil$.

Proof. Let $\mathsf{Img}(g(x)) := \{i : g(x) \in F(i)\}$ be a string function (it is Σ_0^B-definable in $\mathbf{V}^0$). By assumption

$$\forall z \in [t]\, (numones(\mathsf{Img}(g(z))) \leq d)\,. \tag{A.21}$$

Since for any $i \in [t]$, $g(i) \in F(i)$, we can prove in $\mathbf{VTC}^0$ that $\bigcup_{z \in [t]} \mathsf{Img}(g(z))$ equals $\{1, 2, \ldots, t\}$, and so $\mathbf{VTC}^0$ proves:

$$numones\left(\bigcup_{z \in [t]} \mathsf{Img}(g(z))\right) = t. \tag{A.22}$$

Claim A.38. (Under the assumptions of the proposition) $\mathbf{VTC}^0$ proves:

$$numones\left(\bigcup_{z\in[t]}\mathsf{Img}(g(z))\right) \le d\cdot numones(\{g(i)\ :\ i\in[t]\}).$$

Proof of claim: The proof follows from (A.21), by induction on t.
Base case: $t=1$. We have

$$\begin{aligned} numones(\cup_{z\in[t]}\mathsf{Img}(g(z))) &= numones(\mathsf{Img}(g(1))) \\ &\le d && \text{(by assumption)} \\ &= d\cdot numones(\{g(1)\}) \\ &= d\cdot numones(\{g(i)\ :\ i\in[t]\}). \end{aligned}$$

Induction step:
Case 1: $g(t)\in\{g(i)\ :\ i\in[t-1]\}$. Thus,

$$\{g(i)\ :\ i\in[t-1]\} = \{g(i)\ :\ i\in[t]\} \qquad \text{and} \qquad \bigcup_{i\in[t-1]}\mathsf{Img}(g(i)) = \bigcup_{i\in[t]}\mathsf{Img}(g(i)). \tag{A.23}$$

Therefore,

$$\begin{aligned} numones\left(\bigcup_{i\in[t]}\mathsf{Img}(g(i))\right) &= numones\left(\bigcup_{i\in[t-1]}\mathsf{Img}(g(i))\right) \\ &\le d\cdot numones\left(\{g(i)\ :\ i\in[t-1]\}\right) && \text{(by hypothesis)} \\ &= d\cdot numones\left(\{g(i)\ :\ i\in[t]\}\right) && \text{(by (A.23)).} \end{aligned}$$

Case 2: $g(t)\notin\{g(i)\ :\ i\in[t-1]\}$. Thus,

$$numones(\{g(i)\ :\ i\in[t-1]\})+1 = numones(\{g(i)\ :\ i\in[t]\}. \tag{A.24}$$

We have

$$numones\left(\bigcup_{z\in[t]}\mathsf{Img}(g(z))\right) \le numones\left(\bigcup_{z\in[t-1]}\mathsf{Img}(g(z))\right) + numones\left(\mathsf{Img}(g(t))\right),$$

and by induction hypothesis

$$\begin{aligned} &\le d\cdot numones\left(\{g(i)\ :\ i\in[t-1]\}\right) \\ &\qquad + numones\left(\mathsf{Img}(g(t))\right) \\ &\le d\cdot(numones(\{g(i)\ :\ i\in[t]\})-1) \\ &\qquad + numones\left(\mathsf{Img}(g(t))\right) && \text{(by (A.24))} \\ &\le d\cdot(numones(\{g(i)\ :\ i\in[t]\})-1)+d \\ & && \text{(by assumption)} \\ &= d\cdot numones(\{g(i)\ :\ i\in[t]\}). \end{aligned}$$

■Claim

Thus, by Claim A.38 and by (A.22), we get:

$$t \leq d \cdot numones(\{g(i) \ : \ i \in [t]\}),$$

which leads to $t/d \leq numones(\{g(i) \ : \ i \in [t]\})$, and since $numones(\{g(i) \ : \ i \in [t]\})$ is an integer number we get:

$$\lceil t/d \rceil \leq \lceil numones(\{g(i) \ : \ i \in [t]\}) \rceil = numones(\{g(i) \ : \ i \in [t]\}).$$

□

Manipulating big sums in $\mathbf{VTC}^0$

We need to prove basic properties of summation (having a non-constant number of summands) like commutativity, associativity, distributivity, substitution in big sums, rearranging etc., in $\mathbf{VTC}^0$, to be able to carry out basic calculations in the theory. As a consequence of this section we will be able to freely derive inequalities and equalities between big summations (using rearranging, substitutions of equals, etc.) in $\mathbf{VTC}^0$.

Proposition A.39 (Basic properties of sums in $\mathbf{VTC}^0$)**.** In what follows we consider the theory $\mathbf{VTC}^0$ over an extended language (including possibly new Σ_1^B-definable function symbols in $\mathbf{VTC}^0$ and their defining axioms). The function $f(i)$ is a number function symbol mapping to the rationals or naturals (possibly with additional undisplayed parameters). The theory $\mathbf{VTC}^0$ proves the following statements:

Substitution: Assume that $u(i), v(i)$ are two terms (possibly with additional undisplayed parameters), such that $u(i) = v(i)$ for any $i \leq n$, then

$$\sum_{i=0}^{n} f(u(i)) = \sum_{i=0}^{n} f(v(i)).$$

Distributivity: Assume that u is a term that does not contain the variable i, then

$$u \cdot \sum_{i=0}^{n} f(i) = \sum_{i=0}^{n} u \cdot f(i).$$

Rearranging: Assume that $I = \{0, \ldots, n\}$ and let $I_1, \ldots, I_k$ be a definable partition of I (specifically, the sets $I_1, \ldots, I_k$ are all Σ_0^B-definable in $\mathbf{VTC}^0$ and $\mathbf{VTC}^0$ proves that the I_j's form a partition of I). Then

$$\sum_{i=0}^{n} f(i) = \sum_{j=1}^{k} \sum_{i \in I_j} f(i),$$

where $\sum_{i \in I_j} f(i)$ denotes the term $\sum_{i=0}^{|I_j|-1} f(\delta(i))$ where $\delta(i)$ is the function that enumerates (in ascending order) the elements in I_j.

Inequalities: Let $g(i)$ be a number function mapping to the rationals or naturals (possibly with additional undisplayed parameters), such that $f(i) \leq g(i)$ for all $0 \leq i \leq n$, then

$$\sum_{i=0}^{n} f(i) \leq \sum_{i=0}^{n} g(i).$$

Proof.
Substitution: When we work in the theory $\mathbf{VTC}^0$ we implicitly assume that we have equality axioms stating that if $t = t'$, for any two terms t, t', then $F(t) = F(t')$, for any function F (including functions F that are from the extended language of $\mathbf{VTC}^0$). Since we assume that $f(i)$ is a Σ_1^B-definable number function in $\mathbf{VTC}^0$, the function $g(n) := \sum_{i=0}^{n} f(i)$ is also Σ_1^B-definable in $\mathbf{VTC}^0$, and so we also have the equality axiom for $g(n)$. Thus, if $u(i) = v(i)$, for any $i \leq n$, then we can prove also $g(u(n)) = g(v(n))$.

Distributivity: This is proved simply by induction on n. We omit the details.

Rearranging: Because $I_1, \ldots, I_k$ are Σ_0^B-definable sets in $\mathbf{VTC}^0$ we can define the family of sequences $S_1, \ldots, S_k$, each of length $n + 1$, such that

$$S_j[i] := \begin{cases} f(i), & i \in I_j \ ; \\ 0, & \text{otherwise.} \end{cases}$$

The theory $\mathbf{VTC}^0$ proves, by induction on n, that

$$\sum_{j=1}^{k} \sum_{i=0}^{n} S_j[i] = \sum_{i=0}^{n} f(i).$$

For any $j = 1, \ldots, k$, we can Σ_1^B-define in $\mathbf{VTC}^0$ the function $\delta_j : \{0, \ldots, |I_j| - 1\} \to \{0, \ldots, n\}$ such that $\delta_j(\ell) = i$ iff i is the $(\ell + 1)$th element in I_j (when the elements in I_j are ordered in ascending order). In other words, the δ_j's functions enumerate the elements in I_j.

We can now prove in $\mathbf{VTC}^0$ that

$$\sum_{i=0}^{n} S_j[i] = \sum_{i=0}^{|I_j|-1} f(\delta_j(i)),$$

from which, by Substitution (proved above), we can prove:

$$\sum_{i=1}^{k} \sum_{i=0}^{n} S_j[i] = \sum_{i=1}^{k} \sum_{i=0}^{|I_j|-1} f(\delta_j(i)).$$

Inequalities: This can be proved in $\mathbf{VTC}^0$ simply by induction on n. We omit the details. □

All the equalities and inequalities which contain big summations that we will derive in the theory, can be proved using Proposition A.39. We shall not state this explicitly in the text, but continue to derive such equalities and inequalities freely.

The relation between $\mathbf{VTC}^0$ and TC^0-Frege

In this section we show how one can translate a Σ^B_0 formula φ into a family of propositional formulas $[\![\varphi]\!]$. We then state the theorem showing that if the universal closure of a Σ^B_0 formula φ is provable in $\mathbf{VTC}^0$ then the propositional translation $[\![\varphi]\!]$ has a polynomial-size proof in TC^0-Frege.

Definition A.40 (Propositional translation $[\![\cdot]\!]$ of Σ^B_0 formulas)**.** Let $\varphi(\vec{x}, \vec{X})$ be a Σ^B_0 formula. The *propositional translation* of φ is a family

$$[\![\varphi]\!] = \{[\![\varphi]\!]_{\vec{m};\vec{n}} \mid m_i, n_i \in \mathbb{N}\}$$

of propositional formulas in variables $p^{X_i}_j$ for every $X_i \in \vec{X}$. The intended meaning is that $[\![\varphi]\!]$ is a valid family of formulas if and only if the formula

$$\forall \vec{x} \forall \vec{X} \left((\bigwedge |X_i| = \underline{n_i}) \to \varphi(\underline{\vec{m}}, \vec{X}) \right)$$

is true in the standard model $\mathbb{N}_2$ of two sorted arithmetic, where $\underline{n}$ denotes the nth numeral, for any $n \in \mathbb{N}$.

For given $\vec{m}, \vec{n} \in \mathbb{N}$ we define $[\![\varphi]\!]$ by induction on the size of the formula $[\![\varphi]\!]_{\vec{m};\vec{n}}$. We denote the value of a term t by $\mathsf{val}(t)$.

Case 1: Let $\varphi(\vec{x}, \vec{X})$ be an atomic formula.

- If $\varphi(\vec{x}, \vec{X})$ is $\top$ (or $\bot$), then $[\![\varphi]\!]_{\vec{m},\vec{n}} := \top$ (or $\bot$).
- If $\varphi(\vec{x}, \vec{X})$ is $X_i = X_i$, then $[\![\varphi]\!]_{\vec{m},\vec{n}} := \top$.
- If $\varphi(\vec{x}, \vec{X})$ is $X_i = X_j$ for $i \neq j$, then (using the fact that $\mathbf{V}^0$contains the extensionality axiom **SE**) instead of translating φ, we translate the $\mathbf{V}^0$-equivalent formula

$$|X_i| = |X_j| \wedge \forall k \leq |X| \, (X_i(k) \leftrightarrow X_j(k))).$$

- If $\varphi(\vec{x}, \vec{X})$ is $t_1(\vec{y}, |\vec{Y}|) = t_2(\vec{z}, |\vec{Z}|)$ for terms t_1, t_2, number variables $\vec{y}, \vec{z}$ and string variables $\vec{Y}, \vec{Z}$, where $\vec{y} \cup \vec{z} = \vec{x}$ and $\vec{Y} \cup \vec{Z} = \vec{X}$, and $\underline{\vec{m}^y}, \underline{\vec{m}^z}$ and $\underline{\vec{n}^Y}, \underline{\vec{n}^Z}$ denote the corresponding assignments of numerals $\underline{\vec{m}}, \underline{\vec{n}}$ to the $\vec{y}, \vec{z}$ and $\vec{Y}, \vec{Z}$ variables, respectively. Then

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \begin{cases} \top & \text{if } \mathsf{val}(t_1(\underline{\vec{m}^Y}, \underline{\vec{n}^Y})) = \mathsf{val}(t_2(\underline{\vec{m}^Z}, \underline{\vec{n}^Z})) \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

- If $\varphi(\vec{x}, \vec{X})$ is $t_1(\vec{y}, |\vec{Y}|) \leq t_2(\vec{z}, |\vec{Z}|)$ for terms t_1, t_2, number variables $\vec{y}, \vec{z}$ and string variables $\vec{Y}, \vec{Z}$, then

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \begin{cases} \top & \text{if } \mathsf{val}(t_1(\underline{\vec{m}^Y}, \underline{\vec{n}^Y})) \leq \mathsf{val}(t_2(\underline{\vec{m}^Z}, \underline{\vec{n}^Z})) \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

- If $\varphi(\vec{x}, \vec{X})$ is $X_i(t(\vec{x}, |\vec{X}|))$, then

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \bot \text{ if } n_i = 0$$

and otherwise

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \begin{cases} p^{X_i}_{\mathsf{val}(t(\underline{\vec{m}},\underline{\vec{n}}))} & \text{if } \mathsf{val}(t(\underline{\vec{m}},\underline{\vec{n}})) < \underline{n_i - 1}, \\ \top & \text{if } \mathsf{val}(t(\underline{\vec{m}},\underline{\vec{n}})) = \underline{n_i - 1}, \\ \bot & \text{if } \mathsf{val}(t(\underline{\vec{m}},\underline{\vec{n}})) > \underline{n_i - 1}. \end{cases}$$

Case 2: The formula φ is not atomic.

- If $\varphi \equiv \psi_1 \wedge \psi_2$ we let

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := [\![\psi_1]\!]_{\vec{m},\vec{n}} \wedge [\![\psi_2]\!]_{\vec{m},\vec{n}}.$$

- If $\varphi \equiv \psi_1 \vee \psi_2$ we let

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := [\![\psi_1]\!]_{\vec{m},\vec{n}} \vee [\![\psi_2]\!]_{\vec{m},\vec{n}}.$$

- If $\varphi \equiv \neg\psi$ we let

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \neg[\![\psi]\!]_{\vec{m},\vec{n}}.$$

- If $\varphi \equiv \exists y \leq t(\vec{x}, |\vec{X}|)\psi(y, \vec{x}, \vec{X})$ then

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \bigvee_{i=0}^{\mathsf{val}(t(\underline{\vec{m}},\underline{\vec{n}}))} [\![\psi(\underline{i}, \vec{x}, \vec{X})]\!]_{\vec{m},\vec{n}}.$$

- If $\varphi \equiv \forall y \leq t(\vec{x}, |\vec{X}|)\psi(y, \vec{x}, \vec{X})$ then

$$[\![\varphi]\!]_{\vec{m},\vec{n}} := \bigwedge_{i=0}^{\mathsf{val}(t(\underline{\vec{m}},\underline{\vec{n}}))} [\![\psi(\underline{i}, \vec{x}, \vec{X})]\!]_{\vec{m},\vec{n}}.$$

This concludes the translation for Σ^B_0 formulas.

Proposition A.41 (Lemma VII.2.2 [29])**.** For every Σ^B_0 formula $\varphi(\vec{x}, \vec{X})$ there exists a constant $d \in \mathbb{N}$ and a polynomial $p(\vec{m}, \vec{n})$ such that for all $\vec{m}, \vec{n} \in \mathbb{N}$, the propositional translation $[\![\varphi(\vec{x}, \vec{X})]\!]_{\vec{m},\vec{n}}$ has depth at most d and size at most $p(\vec{m}, \vec{n})$.

We can now state the relation between provability of an arithmetical statement φ in $\mathbf{VTC}^0$ to the provability of the family $[\![\varphi]\!]$ in TC^0-Frege as follows.

Theorem A.42 (Section X.4.3. [29])**.** *Let $\varphi(\vec{x}, \vec{X})$ be a Σ^B_0 formula. Then, if $\mathbf{VTC}^0$ proves $\varphi(\vec{x}, \vec{X})$ then there is a polynomial size family of TC^0-Frege proofs of $[\![\varphi]\!]$.*

A.3 Feige-Kim-Ofek Witnesses and the Main Formula

In this section we define the main formula we are going to prove in the theory. We are concerned with proofs of 3CNF formulas. Let us fix the following notation. With n we will denote the number of propositional variables $x_1, \ldots, x_n$ and with m we will denote the number of clauses appearing in the 3CNF denoted $\mathbf{C} = \bigwedge_{\alpha=0}^{m-1} C_\alpha$. Each clause C_α is of the form $x_i^{\ell_1} \vee x_j^{\ell_2} \vee x_k^{\ell_3}$, for $\ell_1, \ell_2, \ell_3 \in \{0,1\}$, where x_i^1 abbreviates x_i and x_i^0 abbreviates $\neg x_i$. A clause C_α is represented by the sequence $\langle i, j, k, \langle \ell_1, \ell_2, \ell_3 \rangle, \alpha \rangle$. The defining Σ_0^B formula of the relation is:

$$\begin{aligned}&\textsc{Clause}(x, n, m) \leftrightarrow \exists i, j, k \leq n \exists \alpha < m \exists \ell \leq 8 \\ &\quad (i > 0 \wedge j > 0 \wedge k > 0 \wedge \langle x \rangle_1^5 = i \wedge \langle x \rangle_2^5 = j \wedge \langle x \rangle_3^5 = k \wedge \langle x \rangle_4^5 = \ell \wedge \langle x \rangle_5^5 = \alpha).\end{aligned}$$

A 3CNF $\mathbf{C} \equiv \bigwedge_{\alpha=0}^{m-1} C_\alpha$ is represented by the sequence $(C_0, \ldots, C_{m-1})$. Since m is non-constant, we use a string variable to code $\mathbf{C}$. The defining Σ_0^B formula of this relation is

$$3\text{CNF}(\mathbf{C}, n, m) \leftrightarrow \forall i < m \, \left(\textsc{Clause}(\mathbf{C}[i], n, m) \wedge \langle \mathbf{C}[i] \rangle_5^5 = i \right).$$

For a number variable x, we Σ_0^B-define $\textsc{Even}(x)$ by the formula $\exists y \leq x (2 \cdot y = x)$ (meaning that x is an even number). Accordingly, we define $\textsc{Odd}(x)$ by $\neg\textsc{even}(x)$.

For some clause C and a string variable A (interpreted as a Boolean assignment), we Σ_0^B-define the following predicate, stating that C is not satisfied under the assignment A:

$$\begin{aligned}\textsc{NotSAT}(C, A) \equiv &\exists i, j, k \leq n \\ &\left(\langle C \rangle_1^5 = i \wedge (A(i) \leftrightarrow \langle \langle C \rangle_4^5 \rangle_1^3 = 0) \right) \\ &\wedge \left(\langle C \rangle_2^5 = j \wedge (A(j) \leftrightarrow \langle \langle C \rangle_4^5 \rangle_2^3 = 0) \right) \\ &\wedge \left(\langle C \rangle_3^5 = k \wedge (A(k) \leftrightarrow \langle \langle C \rangle_4^5 \rangle_3^3 = 0) \right).\end{aligned}$$

We need the following notations and definitions to facilitate the formalization of certain sets and objects:

Notation:

1. When considering a *set* of clauses, then a clause in $\mathbf{C}$ will be referred to only by its index $0 \leq i < m$. Thus, a set of clauses from $\mathbf{C}$ is a set of natural numbers less than m.

2. A set of literal positions from $\mathbf{C}$ will be coded as a set of numbers $\langle a, b \rangle$, where $0 \leq a < m$ is the index of a clause in $\mathbf{C}$ and $b = 1, 2, 3$ is the index of a literal in the clause.

3. For $0 \leq i < m$ and $\varepsilon = 0, 1$ and a sequence S of 3-clauses we define $\mathsf{LitPos}(S, i, \varepsilon)$ to be the string function that outputs the set of (positions of) literals x_i^ε in S. In other words, we have:

   $$\mathsf{LitPos}(S, i, \varepsilon) := \left\{ \langle j, \ell \rangle \ : \ j < \mathit{length}(S) \wedge \ell \leq 3 \wedge \langle S[j] \rangle_\ell^5 = i \ \wedge \ \langle \langle S[j] \rangle_4^5 \rangle_\ell^3 = \varepsilon \right\}.$$

4. Let $\mathsf{satLit}(A, \mathbf{C})$ be the string function that outputs the set of all literal positions in $\mathbf{C}$ that are satisfied by A.

5. The function $\mathsf{Lit}(C, i)$ returns the ith literal x_j^ε of the clause C, for $i = 1, 2, 3$, in the form of a pair $\langle j, \varepsilon \rangle$.

6. If the literals of a clause are not all true or not all false under A, then we say that the clause is satisfied as NAE (standing for "not all equal") by A. We can easily Σ_0^B-define the predicate $\textsc{SatL}(z, A)$, stating that the literal z is satisfied by the assignment A in $\mathbf{VTC}^0$. Let:

$$\begin{aligned}\mathrm{NAE}(C, A) \leftrightarrow \textsc{Clause}(C) \wedge \bigvee_{i=1,2,3} \textsc{SatL}\left(\mathsf{Lit}(C, i), A\right) \wedge \\ \bigvee_{i=1,2,3} \neg\textsc{SatL}\left(\mathsf{Lit}(C, i), A\right)\end{aligned}$$

   be the Σ_0^B relation that states that the assignment A satisfies the 3-clause C as NAE. Let $\mathsf{satNAE}(A, \mathbf{C})$ be the string function that outputs the set of clauses in $\mathbf{C}$ that are satisfied as NAE by A.

The functions $\mathsf{LitPos}(S, i, \varepsilon)$, $\mathsf{satLit}(A, \mathbf{C})$ and $\mathsf{satNAE}(A, \mathbf{C})$ above are all $\mathbf{AC}^0$-reducible to the language $\mathcal{L}_A^2$ and so we can assume that we have these functions (along with their defining axioms) in $\mathbf{VTC}^0$ (see Section A.2.1). All the functions in this section will be $\mathbf{AC}^0$-reducible to $\mathcal{L}_A^2 \cup \{numones\}$, and all the relations in this section will have Σ_0^B definitions in the language $\mathcal{L}_A^2$ extended to include both our new function symbols and *numones*.

Definition A.43 (Even k-tuple)**.** For any given k, a sequence S of k clauses is an *even k-tuple* iff every variable appears an even number of times in the sequence. Formally, the predicate is denoted $\mathrm{TPL}(S, k)$:

$$\begin{aligned}\mathrm{TPL}(S, k) \leftrightarrow &\mathit{length}(S) = k \wedge \\ &\forall i \leq n,\ \textsc{Even}\left(\mathit{numones}(\mathsf{LitPos}(S, i, 0)) + \mathit{numones}(\mathsf{LitPos}(S, i, 1))\right).\end{aligned} \tag{A.25}$$

Observe that if S is an even k-tuple then k is even (since the total number of variable occurrences N is even, by assumption that each variable occurs an even number of times; and $k = N/3$, since each clause has three variables).

Definition A.44 (Inconsistent k-tuple)**.** An even k-tuple is said to be *inconsistent* if the total number of negations in its clauses is odd. Formally, the predicate is denoted by $\mathrm{ITPL}(S, k)$:

$$\mathrm{ITPL}(S, k) \leftrightarrow \mathrm{TPL}(S, k) \wedge \textsc{Odd}\left(\sum_{i=1}^{n} \mathit{numones}(\mathsf{LitPos}(S, i, 1))\right).$$

Definition A.45 (The imbalance $\textsc{Imb}(S, y)$)**.** For a 3CNF S we define the function *i-imbalance* $\mathsf{ilmb}(S, i)$ to be the absolute value of the difference of negated occurrences of x_i and non-negated occurrences of x_i in the 3CNF S (where

$x_1, \ldots, x_n$ are considered to be all the variables in S). It is defined simply by the term:

$$\mathsf{iImb}(S, i) := abs(numones(\mathsf{LitPos}(i, 0, S)) - numones(\mathsf{LitPos}(i, 1, S))).$$

For a 3CNF S, the predicate *imbalance of* S, denoted $\textsc{Imb}(S, y)$, is true iff y equals the sum over the i-imbalances of all the variables, that is:

$$\textsc{Imb}(S, y) \leftrightarrow y = \sum_{i=1}^{n} \mathsf{iImb}(S, i).$$

Definition A.46 ((t, k, d)-collection)**.** A (t, k, d)-*collection* $\mathscr{D}$ of a 3CNF $\mathbf{C}$ with m clauses is an array (coded as in Definition A.27) of t many inconsistent k-tuples, which contain only clauses from $\mathbf{C}$, and each clause appears in at most d many such inconsistent k-tuples. The predicate is denoted $\textsc{Coll}(t, k, d, \mathbf{C}, \mathscr{D})$ and is defined by the following formula:

$$\begin{aligned} &length(\mathscr{D}) = t \wedge \\ &\quad \forall i < t \, \mathrm{ITPL}(\mathscr{D}^{[i]}, k) \wedge \\ &\quad \forall i < t \forall \ell < k \exists j < |\mathbf{C}| \, (\mathscr{D}^{[i]}[\ell] = \mathbf{C}[j]) \wedge \\ &\quad \forall j < |\mathbf{C}| \sum_{i=0}^{t-1} \sum_{\ell=0}^{k-1} \chi_{=}(\langle \mathscr{D}^{[i]}[\ell] \rangle_5^5, j) \leq d. \end{aligned}$$

Definition A.47 ($\textsc{Mat}(M, \mathbf{C})$)**.** We define the predicate $\textsc{Mat}(M, \mathbf{C})$ that holds iff M is an $n \times n$ rational matrix such that M_{ij} equals $\frac{1}{2}$ times the number of clauses in $\mathbf{C}$ where x_i and x_j appear with different polarity minus $\frac{1}{2}$ times the number of clauses where they appear with the same polarity. More formally, we have

$$M_{ij} := \sum_{k=0}^{m-1} E_{ij}^{(k)}, \qquad \text{for any } i, j \in [n], \tag{A.26}$$

where $E_{ij}^{(k)}$ corresponds to the kth clause in $\mathbf{C}$ as follows:

$$E_{ij}^{(k)} := \begin{cases} \frac{1}{2}, & x_i^{\varepsilon_i}, x_j^{\varepsilon_j} \in \mathbf{C}[k] \text{ and } \varepsilon_i \neq \varepsilon_j, \text{ for some } \varepsilon_i, \varepsilon_j \in \{0, 1\} \text{ and } i \neq j; \\ -\frac{1}{2}, & x_i^{\varepsilon_i}, x_j^{\varepsilon_j} \in \mathbf{C}[k] \text{ and } \varepsilon_i = \varepsilon_j, \text{ for some } \varepsilon_i, \varepsilon_j \in \{0, 1\} \text{ and } i \neq j; \\ 0, & \text{otherwise.} \end{cases} \tag{A.27}$$

Note that $E_{ij}^{(k)}$ is definable by a Σ_0^B formula (in $\mathcal{L}_A^2$), and so $\textsc{Mat}(M, \mathbf{C})$ is a Σ_0^B-definable relation in $\mathbf{VTC}^0$.

Finally, we need a predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$ that ensures that $\vec{\lambda}$ is a collection of n rational approximations of the eigenvalues of the matrix M and that V is the rational matrix whose rows are the rational approximations of the eigenvectors of M (where the ith row in V is the approximation of the approximate eigenvector λ_i). For the sake of readability we defer the formal definition of the predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$ and all the lemmas that relate to it, including the proofs in the theory making use of this predicate, to Section A.5.

Notation: 1. The notation $o(1)$ appearing inside a formula in the proof within the theory, and specifically in Definition A.48 below, stands for a term of the

form b/n^c, for b a number symbol greater than 0, and c some positive constant (and where a rational number is encoded in the way described in Section A.2.1).

2. Given two terms t and $f(n)$ in the language $\mathcal{L}_A^2$, where n is a number variable, we say that $\mathbf{VTC}^0$ *proves* $t = O(f(n))$, to mean that there exists some constant c (independent of n) such that $\mathbf{VTC}^0$ proves $t \leq c \cdot f(n)$, where c is a term without variables in the language $\mathcal{L}_A^2$.

We can now state the main formula that we are going to prove in $\mathbf{VTC}^0$. It says that if the Feige-Kim-Ofek witness fulfills the inequality $t > \frac{d\cdot(I+\lambda n)}{2} + o(1)$ then there exists a clause in $\mathbf{C}$ that is not satisfied by any assignment A (one can think of all the free variables in the formula as universally quantified):

Definition A.48 (The main formula)**.** The *main formula* is the following formula ($\vec{\lambda}$ denotes n distinct number parameters $\lambda_1, \ldots, \lambda_n$):

$$\Big(3\mathrm{CNF}(\mathbf{C}, n, m) \wedge \textsc{Coll}(t, k, d, \mathbf{C}, \mathscr{D}) \wedge \textsc{Imb}(\mathbf{C}, I) \wedge \textsc{Mat}(M, \mathbf{C}) \wedge$$
$$\textsc{EigValBound}(M, \vec{\lambda}, V) \wedge \lambda = \max\{\lambda_1, \ldots, \lambda_n\} \wedge t > \frac{d \cdot (I + \lambda n)}{2} + o(1)\Big)$$
$$\longrightarrow \exists i < m\, \textsc{NotSAT}(\mathbf{C}[i], A).$$

A.4 Proof of the Main Formula

In this section we prove our key theorem:

Theorem A.49 (Key)**.** *The theory* $\mathbf{VTC}^0$ *proves the main formula (Definition A.48).*

Proof. We reason inside $\mathbf{VTC}^0$. Assume by way of contradiction that the premise of the implication in the main formula holds and that there is an assignment $A \in \{0,1\}^n$ (construed as a string variable of length n) that satisfies every clause in $\mathbf{C}$. Recall that $\mathsf{satLit}(A, \mathbf{C})$ is the set of all literal positions that are satisfied by A.

Lemma A.50. *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves:*

$$\mathit{numones}(\mathsf{satLit}(A, \mathbf{C})) \leq \frac{3m + I}{2}.$$

Proof. First observe that for any assignment A and any $1 \leq i \leq n$ the set of satisfied literals of x_i is defined by $\mathsf{LitPos}(\mathbf{C}, i, A(i))$. Therefore, the sets $\mathsf{LitPos}(\mathbf{C}, 1, A(1)), \ldots, \mathsf{LitPos}(\mathbf{C}, n, A(n))$ form a partition of $\mathsf{satLit}(A, \mathbf{C})$ (provably in $\mathbf{VTC}^0$), and thus by Proposition A.34, $\mathbf{VTC}^0$ proves that

$$\mathit{numones}(\mathsf{satLit}(A, \mathbf{C})) = \sum_{i=1}^{n} \mathit{numones}(\mathsf{LitPos}(\mathbf{C}, i, A(i))). \tag{A.28}$$

By (A.28) we get

$$numones(\mathsf{satLit}(A,\mathbf{C})) \leq \sum_{i=1}^{n} \max\{numones(\mathsf{LitPos}(\mathbf{C},i,0)), numones(\mathsf{LitPos}(\mathbf{C},i,1))\}. \tag{A.29}$$

For any $1 \leq i \leq n$, define the term

$$\mathsf{LitPos}(\mathbf{C},i) := \mathsf{LitPos}(\mathbf{C},i,0) \cup \mathsf{LitPos}(\mathbf{C},i,1).$$

Then by

$$\frac{\mathsf{ilmb}(\mathbf{C},i) + numones(\mathsf{LitPos}(\mathbf{C},i))}{2} = \frac{\mathsf{ilmb}(\mathbf{C},i) + numones(\mathsf{LitPos}(\mathbf{C},i,0)) + numones(\mathsf{LitPos}(\mathbf{C},i,1))}{2},$$

and since, by Definition A.45,
$\mathsf{ilmb}(\mathbf{C},i) = abs\,(numones(\mathsf{LitPos}(\mathbf{C},i,0)) - numones(\mathsf{LitPos}(\mathbf{C},i,1)))$, the theory $\mathbf{VTC}^0$ proves that for any $1 \leq i \leq n$:

$$\max\{numones(\mathsf{LitPos}(\mathbf{C},i,0)), numones(\mathsf{LitPos})(\mathbf{C},i,1)\} = \frac{\mathsf{ilmb}(\mathbf{C},i) + numones(\mathsf{LitPos}(\mathbf{C},i))}{2}. \tag{A.30}$$

Claim A.51. (Assuming the premise of the main formula) the theory $\mathbf{VTC}^0$ proves:

$$\sum_{i=1}^{n} \frac{\mathsf{ilmb}(\mathbf{C},i) + numones(\mathsf{LitPos}(\mathbf{C},i))}{2} = \frac{I+3m}{2}.$$

Proof of claim: First recall the definition of imbalance (Definition A.45) $I = \sum_{i=1}^{n} \mathsf{ilmb}(\mathbf{C},i)$. Thus it remains to prove that $\sum_{i=1}^{n} numones(\mathsf{LitPos}(\mathbf{C},i)) = 3m$. For this, note that $\mathsf{LitPos}(\mathbf{C},i)$, for $i = 1,\ldots,n$, partition the set of all literal positions in $\mathbf{C}$. In other words, we can prove that: (i) if H is the set of all literal positions in $\mathbf{C}$ (this set is clearly Σ_0^B-definable in $\mathbf{VTC}^0$) then $H = \cup_{i=1}^{n}\mathsf{LitPos}(\mathbf{C},i)$; and (ii) $\mathsf{LitPos}(\mathbf{C},i) \cap \mathsf{LitPos}(\mathbf{C},j) = \emptyset$, for all $1 \leq i \neq j \leq n$. Therefore, by Proposition A.34 we can prove that:

$$numones(H) = \sum_{i=1}^{n} numones(\mathsf{LitPos}(\mathbf{C},i)). \tag{A.31}$$

Now, the set H of all literal position in $\mathbf{C}$ can be partitioned (provably in $\mathbf{VTC}^0$) by the sets $T_1,\ldots,T_m$, where each T_j, for $0 \leq j < m$, is the set of the three literals in the jth clause in $\mathbf{C}$. Thus, again by Proposition A.34, we can prove that $numones(H) = 3m$. By (A.31) we therefore have

$$\sum_{i=1}^{n} numones(\mathsf{LitPos}(\mathbf{C},i)) = 3m.$$

■Claim

We conclude that:

$$
\begin{aligned}
&\mathit{numones}(\mathsf{satLit}(A,\mathbf{C})) \\
&\quad\le \sum_{i=1}^{n} \max\{\mathit{numones}(\mathsf{LitPos}(\mathbf{C},i,0)), \mathit{numones}(\mathsf{LitPos}(\mathbf{C},i,1)) && \text{(by (A.29))} \\
&\quad= \sum_{i=1}^{n} \frac{\mathsf{iImb}(\mathbf{C},i)+\mathsf{LitPos}(\mathbf{C},i)}{2} && \text{(by (A.30))} \\
&\quad= \frac{I+3m}{2}. && \text{(by Claim A.51).}
\end{aligned}
$$

□

We now bound the number of clauses in $\mathbf{C}$ that contain exactly two literals satisfied by A. We say that a 3-clause is satisfied by a given assignment as NAE (which stands for *not all equal*) if the literals in the clause do not all have the same truth values. That is, if either exactly one or exactly two literals in the clause are satisfied by the assignment.

Recall that $\mathsf{satNAE}(A,\mathbf{C})$ is the function that returns the set of all clauses (formally, indices $< m$) that are satisfied as NAE by A.

Lemma A.52. *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves: let* h *be the number of clauses in* $\mathbf{C}$ *that contain exactly two literals satisfied by* A*. Then*

$$h \le \frac{3m+I}{2} - 3m + 2\cdot \mathit{numones}(\mathsf{satNAE}(A,\mathbf{C}))\,.$$

Proof. For $i=0,1,2,3$, let B_i be the set of clauses in $\mathbf{C}$ that contain exactly i literals satisfied by A. For $i=0,1,2,3$, let F_i be the string function that maps a clause (index) C to the set of literal positions that are satisfied by A in case there are exactly i such literals and to the empty set otherwise:

$$F_i(j) = \begin{cases} \{l_1,\ldots,l_i\}, & \text{if } j\in B_i\ ; \\ \emptyset, & \text{otherwise} \end{cases}$$

(where a literals l_k is coded, as before, by the pair $\langle a,b\rangle$ for a an index of a clause in $\mathbf{C}$ and b the position of the literal in the clause). Every such function F_i is Σ^B_0-defined in $\mathbf{VTC}^0$. We also Σ^B_0-define the image of F_i as follows:

$$\mathrm{Img}(F_i) := \{x :\ \exists y < m\,(F_i(y))(x)\}.$$

Claim A.53. (Assuming the premise of the main formula) the theory $\mathbf{VTC}^0$ proves:

$$\mathit{numones}(\mathsf{satLit}(A,\mathbf{C})) = \sum_{i=1}^{3} \mathit{numones}(\mathrm{Img}(F_i)).$$

Proof of claim: In light of Proposition A.34, it suffices to prove that $\mathsf{satLit}(A,\mathbf{C})$ is partitioned by $\mathrm{Img}(F_1),\mathrm{Img}(F_2),\mathrm{Img}(F_3)$ (note that $\mathrm{Img}(F_0)=\emptyset$ by definition), in the sense that:

(i) $\mathsf{satLit}(A,\mathbf{C}) = \mathrm{Img}(F_1)\cup\mathrm{Img}(F_2)\cup\mathrm{Img}(F_3)$, and

(ii) $\mathrm{Img}(F_i) \cap \mathrm{Img}(F_j) = \emptyset$, for all $1 \leq i \neq j \leq 3$.

We prove (i): consider a literal $x \in \mathsf{satLit}(A, \mathbf{C})$, and let $x = \langle a, b \rangle$. We know that the clause C_a contains the literal x. Now, either zero, or one, or two of the remaining literals in C_a are satisfied by A. So x must be in either $F_1(a)$ or in $F_2(a)$ or in $F_3(a)$, respectively. Item (ii) is easy to prove by the definition of the F_i's. We omit the details. $\blacksquare_{\text{Claim}}$

Claim A.54. For any $i = 1, 2, 3$, $numones(\mathrm{Img}(F_i)) = i \cdot numones(B_i)$.

Proof of claim: Fix some $i = 1, 2, 3$. We prove the claim by induction on the number of clauses $j < m$ (we can consider the sets B_i and the functions F_i having an additional parameter that determines until which clause to build the sets. That is, $B_i(z)$ is the set of clauses from 0 to z that have i literals satisfied by A; and similarly we add a parameter for the F_i's). In the base case $j = 0$ there is only one clause C_0. Depending on A we know how many literals in C_0 are satisfied by A. And so $0 \in B_i$ iff i literals are satisfied by A in C_0 iff $numones(F_i(0)) = i = i \cdot 1 = i \cdot numones(B_i)$. The induction step is similar and we omit the details. $\blacksquare_{\text{Claim}}$

By Claim A.53 and Claim A.54 we get:

$$\begin{aligned} numones(\mathsf{satLit}(A, \mathbf{C})) &= \sum_{i=1,2,3} numones(\mathrm{Img}(F_i)) \\ &= \sum_{i=1,2,3} i \cdot numones(B_i)\,. \end{aligned} \tag{A.32}$$

It is easy to show (in a similar manner to Claim A.53) that $B_1 \cup B_2 \cup B_3 = \{0, \ldots, m-1\}$ and $B_i \cap B_j = \emptyset$, for any $1 \leq i \neq j \leq 3$. From this, using Proposition A.34, we get that $m = numones(B_1) + numones(B_2) + numones(B_3)$, and so:

$$numones(B_1) = m - numones(B_2) - numones(B_3)\,. \tag{A.33}$$

Thus, by (A.32):

$$\begin{aligned} numones(\mathsf{satLit}(A, \mathbf{C})) &= m - numones(B_2) - numones(B_3) \\ &\quad + 2 \cdot numones(B_2) + 3 \cdot numones(B_3) \\ &= m + 2 \cdot numones(B_3) + numones(B_2)\,, \end{aligned} \tag{A.34}$$

and so

$$numones(B_2) = numones(\mathsf{satLit}(A, \mathbf{C})) - m - 2 \cdot numones(B_3)\,. \tag{A.35}$$

The set of clauses in $\mathbf{C}$ that are NAE satisfied by A (i.e., $\mathsf{satNAE}(A, \mathbf{C})$) is equal to the set of clauses having either one or two literals satisfied by A; the latter two sets are just B_1 and B_2, and since they are (provably in $\mathbf{VTC}^0$) disjoint we have (using also (A.33)):

$$numones(B_3) = m - (numones(B_1) + numones(B_2)) = m - numones(\mathsf{satNAE}(A, \mathbf{C}))\,.$$

Plugging this into (A.35), and using Lemma A.50, we get:

$$\begin{aligned} numones(B_2) &= numones(\mathsf{satLit}(A, \mathbf{C})) - 3m + 2 \cdot numones(\mathsf{satNAE}(A, \mathbf{C})) \\ &\leq \frac{3m + I}{2} - 3m + 2 \cdot numones(\mathsf{satNAE}(A, \mathbf{C})). \end{aligned}$$

This concludes the proof of Lemma A.52 □

The following lemma provides an upper bound on the number of clauses in $\mathbf{C}$ that can be satisfied as NAE by the assignment A.

Lemma A.55. *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves:*

$$numones(\mathsf{satNAE}(A, \mathbf{C})) \leq (n\lambda + 3m)/4 + o(1).$$

The proof of this lemma involves a spectral argument. Carrying out this argument in the theory is fairly difficult because one has to work with rational approximations (as the eigenvalues and eigenvectors might be irrationals, and so undefined in the theory) and further the proof must be sufficiently constructive, in the sense that it would fit in the theory $\mathbf{VTC}^0$. We thus defer to a separate section (Section A.5) all treatment of the spectral argument. Given the desired spectral inequality, we can prove Lemma A.55—this is done in Section A.4.2.

We can now finish the proof of the key theorem:

Concluding the proof of the theorem (Theorem A.49). In $\mathbf{VTC}^0$ (and assuming the premise of the main formula), let h be the number of clauses in $\mathbf{C}$ that contain exactly two literals satisfied by A. We have:

$$\begin{aligned} h &\leq \frac{3m + I}{2} - 3m + 2 \cdot numones(\mathsf{satNAE}(A, \mathbf{C})) && \text{(by Lemma A.52)} \\ &\leq \frac{3m + I}{2} - 3m + \frac{3m + \lambda n}{2} + o(1) && \text{(by Lemma A.55)} \\ &= \frac{I + \lambda n}{2} + o(1)\,. && \text{(A.36)} \end{aligned}$$

Since we assumed that A satisfies $\mathbf{C}$, then every clause in $\mathbf{C}$ has at least one literal satisfied by A. Thus, the clauses in $\mathbf{C}$ that are not satisfied as 3XOR by A are precisely the clauses that have exactly two literals satisfied by A. By (A.36), the number of clauses that have exactly two literals satisfied by A is at most $\frac{I+\lambda n}{2} + o(1)$. We now use Lemma A.57 (proved in the next subsection) to prove the following lemma:

Lemma A.56. *(Assuming the premise of the main formula) the theory* $\mathbf{VTC}^0$ *proves that the number of clauses in* $\mathbf{C}$ *that are not satisfied as 3XOR by* A *is at least* $\lceil t/d \rceil$.

Proof. Consider the collection $\textsc{Coll}(t, k, d, \mathbf{C}, \mathscr{D})$ in the premise of the main formula. Then, $\mathscr{D}$ is a sequence of t inconsistent k-tuples from $\mathbf{C}$, and every pair of k-tuples in $\mathscr{D}$ intersect[6] on at most d clauses from $\mathbf{C}$. By Lemma A.57, each of the t inconsistent k-tuples contains a clause which is unsatisfied as 3XOR by

[6]Where a clause is identified with its index $0, \ldots, m-1$ in $\mathbf{C}$, so that two identical clauses with a different index are considered as two different clauses.

A. Since each such clause may appear in at most d other inconsistent k tuples, using Proposition A.37 the theory $\mathbf{VTC}^0$ proves that the total number of distinct clauses not satisfied as 3XOR by A is at least $\lceil t/d \rceil$. □

Using this Lemma, we can finish the proof of the key Theorem A.49, as follows: by Lemma A.56 and the fact that the number of clauses in $\mathbf{C}$ that are not satisfied as 3XOR by A is at most $\frac{I+\lambda n}{2} + o(1)$, we get

$$t = d \cdot \frac{t}{d} \leq d \cdot \left\lceil \frac{t}{d} \right\rceil \leq d \cdot \frac{I + \lambda n}{2} + o(1)\,, \tag{A.37}$$

which contradicts our assumption (in the main formula) that $t > \frac{d(I+\lambda n)}{2} + o(1)$. Formally, we need to take care here for the "$o(1)$" notation. Recall that $o(1)$ stands for a term b/n^c for some constants number term b and a constant c. Therefore, it is enough to require that if our assumption (in the premise of the main formula) is $t > \frac{d(I+\lambda n)}{2} + b/n^c$, then in (A.37) above we have $t \leq d \cdot \left\lceil \frac{t}{d} \right\rceil \leq d \cdot \frac{I+\lambda n}{2} + b'/n^{c'}$, so that $b/n^c \leq b'/n^{c'}$. (This requirement will be easily satisfied when applying our theorem (see Corollary A.76).) □

A.4.1 Formulas satisfied as 3XOR

Here we prove the missing lemma that was used in the proof of Lemma A.56.

Notation: For a sequence S of k many 3-clauses, and for $0 \leq \alpha < k$, we denote the three variables in the clause $S[\alpha]$ by $x_{i_\alpha}, x_{j_\alpha}, x_{h_\alpha}$, and abbreviate $\langle\langle S[\alpha]\rangle_4^5\rangle_t^3$, which is the polarity of the tth variable in $S[\alpha]$, by ℓ_t^α, for $t = 1, 2, 3$. Thus, $x_i^{\ell_1^\alpha}, x_j^{\ell_2^\alpha}, x_h^{\ell_3^\alpha}$, are the three literals in $S[\alpha]$ and the values of $\neg A(i) \oplus \ell_1^\alpha, \neg A(j) \oplus \ell_2^\alpha, \neg A(h) \oplus \ell_3^\alpha$ are the values that A assigns to $x_i^{\ell_1^\alpha}, x_j^{\ell_2^\alpha}, x_h^{\ell_3^\alpha}$, respectively, where $\oplus$ is the XOR operator. We also abuse notation and write $\neg A(i)$ inside a term to mean the *characteristic function* of the predicate $\neg A(i)$, that is, the function that returns 1 if $\neg A(i)$ is true, and 0 otherwise.

For a clause C and an assignment A the predicate $\mathrm{3XOR}(C, A)$ says that A satisfies exactly one or three of the literals in C. If we denote by x_i, x_j, x_h the three variables in C and by ℓ_1, ℓ_2, ℓ_3 their respective polarities, we have:

$$\mathrm{3XOR}(C, A) \quad \text{iff} \quad \neg A(i) \oplus \ell_1 + \neg A(j) \oplus \ell_2 + \neg A(h) \oplus \ell_3 = 1 \mod 2\,,$$

and formally the predicate 3XOR is Σ_0^B-definable by the following formula:

$$\mathrm{3XOR}(C, A) := \textsc{Odd}(\neg A(i) + \ell_1 + \neg A(j) + \ell_2 + \neg A(h) + \ell_3)\,.$$

Lemma A.57. *The theory $\mathbf{VTC}^0$ proves that if S is an inconsistent (even) k-tuple, then for every assignment A to its variables there exists $\alpha < k$ such that A satisfies exactly zero or exactly two literals in the clause $S[\alpha]$. More formally, $\mathbf{VTC}^0$ proves:*

$$\forall A \leq n\, \forall k \leq n \forall S \leq p(n)\, \exists \alpha < k\, (|A| = n \wedge \mathrm{ITPL}(S, k) \rightarrow \neg \mathrm{3XOR}\,(S[\alpha], A))\,,$$

for some (polynomial) term $p(\cdot)$.

Proof. We need the following claim:

Claim A.58. Let $f(y)$ be a number function definable in $\mathbf{VTC}^0$. Then $\mathbf{VTC}^0$ proves the following statements:

1. $(\forall \alpha < k, \textsc{Odd}(f(\alpha))) \wedge \textsc{Even}(k) \rightarrow \textsc{Even}\left(\sum_{\alpha=0}^{k-1} f(\alpha)\right)$;
2. $(\forall \alpha < k, \textsc{Even}(f(\alpha))) \rightarrow \textsc{Even}\left(\sum_{\alpha=0}^{k-1} f(\alpha)\right)$;
3. $(\forall \alpha < k, \textsc{Odd}(f(\alpha))) \wedge \textsc{Odd}(k) \rightarrow \textsc{Odd}\left(\sum_{\alpha=0}^{k-1} f(\alpha)\right)$.

Proof of claim: Consider Item 1 (the other items are similar). The proof is by induction on k, showing that

$$((\forall \alpha < k \exists y(2y+1 = f(\alpha))) \wedge \exists y(2y = k)) \rightarrow \exists y \sum_{\alpha=0}^{k-1} f(\alpha) = 2y\,,$$

and using the fact that $\mathbf{V}^0$ proves that $\textsc{Odd}(x) \leftrightarrow \exists y \leq x(2y+1 = x)$ (e.g., by induction on x). We omit the details. $\blacksquare_{\text{Claim}}$

Now, assume by way of contradiction that A satisfies all the clauses in S as 3XORs. Thus, for any $\alpha < k$, if we define $f(\alpha) := \neg A(i_\alpha) + \ell_1^\alpha + \neg A(j_\alpha) + \ell_2^\alpha + \neg A(h_\alpha) + \ell_3^\alpha$, then $\textsc{Odd}(f(\alpha))$. Hence, because $\textsc{Even}(k)$, by Claim A.58 we can prove that:

$$\sum_{\alpha=0}^{k-1} (\neg A(i_\alpha) \oplus \ell_1^\alpha + \neg A(j_\alpha) \oplus \ell_2^\alpha + \neg A(h_\alpha) \oplus \ell_3^\alpha) = 0 \mod 2. \tag{A.38}$$

Recall that every variable appears an even number of times in S. Thus, if a variable has an odd number of negative appearances then it also has an odd number of positive appearances. Similarly, if a variable has an even number of negative appearances then it also has an even number of positive appearances. Let $I_0 \in \{0, \ldots, n-1\}$ be the indices of variables having an even number of positive (and thus negative) appearances in S and let $I_1 = \{0, \ldots, n-1\} \setminus I_0$ be the indices of variables having an odd number of positive (and thus negative) appearances in S. Thus, the left hand side of (A.38), can be written as follows (for $\varepsilon = 0, 1$, we denote by $x_i^\varepsilon(A)$ the truth value of the literal x_i^ε under A):

$$\begin{aligned} &\sum_{i \in I_0} \left(\underbrace{x_i^1(A) + \ldots + x_i^1(A)}_{\text{even times}} + \underbrace{x_i^0(A) + \ldots + x_i^0(A)}_{\text{even times}} \right) + \\ &\sum_{i \in I_1} \left(\underbrace{x_i^1(A) + \ldots + x_i^1(A)}_{\text{odd times}} + \underbrace{x_i^0(A) + \ldots + x_i^0(A)}_{\text{odd times}} \right). \end{aligned} \tag{A.39}$$

Claim A.59. For any $i \in I_0$ (and any string variable A of size n) the theory $\mathbf{VTC}^0$ proves that

$$\underbrace{x_i^1(A) + \ldots + x_i^1(A)}_{\text{even times}} + \underbrace{x_i^0(A) + \ldots + x_i^0(A)}_{\text{even times}}$$

is an even number.

Proof of claim: Reason in $\mathbf{VTC}^0$ as follows: assume that $A(i) = 0$. Then $x_i^1(A) = 0$ and $x_i^0(A) = 1$ and so by Claim A.58 the sum of evenly many $x_i^1(A)$'s is even and the sum of evenly many $x_i^0(A)$'s is also even. The sum of two even numbers is even, and so we are done. (The case where $A(i) = 1$ is similar.) $\blacksquare_{\text{Claim}}$

By Claims A.58 and A.59, the theory $\mathbf{VTC}^0$ proves

$$\textsc{Even}\left(\sum_{i\in I_0}\left(\underbrace{x_i^1(A)+\ldots+x_i^1(A)}_{\text{even times}}+\underbrace{x_i^0(A)+\ldots+x_i^0(A)}_{\text{even times}}\right)\right). \tag{A.40}$$

Similarly to the above claims we have:

Claim A.60. For any $i \in I_1$ (and any string variable A of size n) the theory $\mathbf{VTC}^0$ proves that

$$\underbrace{x_i^1(A)+\ldots+x_i^1(A)}_{\text{odd times}}+\underbrace{x_i^0(A)+\ldots+x_i^0(A)}_{\text{odd times}}$$

is an odd number.

Since by assumption S is an inconsistent k-tuple, the number of negative literals is odd (Definition A.44), and so (provably in $\mathbf{VTC}^0$) the number of variables that has an odd number of negative appearances must be odd, in other words, $|I_1|$ is odd. Therefore, by Claims A.60 and A.58, $\mathbf{VTC}^0$ proves:

$$\textsc{Odd}\left(\sum_{i\in I_1}\left(\underbrace{x_i^1(A)+\ldots+x_i^1(A)}_{\text{odd times}}+\underbrace{x_i^0(A)+\ldots+x_i^0(A)}_{\text{odd times}}\right)\right). \tag{A.41}$$

Since $\mathbf{VTC}^0$ proves both (A.40) and (A.41), $\mathbf{VTC}^0$ proves that (A.39) is odd, which contradicts (A.38). This implies that not all the clauses in S are satisfied as 3XOR by the assignment A. □

A.4.2 Bounding the number of NAE satisfying assignments

Here we prove Lemma A.55 used to prove the key theorem (Theorem A.49). Recall that $\mathsf{satNAE}(A, \mathbf{C})$ is the string function that outputs the set of clauses in $\mathbf{C}$ that are satisfied as NAE by A (see Section A.3). The proof of the following lemma is based on the spectral inequality proved in Section A.5.

Lemma A.55 *(Assuming the premise of the main formula)* $\mathbf{VTC}^0$ *proves*

$$\mathit{numones}(\mathsf{satNAE}(A, \mathbf{C})) \leq (n\lambda + 3m)/4 + o(1).$$

Proof. Let $\mathbf{a}$ be a vector from $\{-1, 1\}^n$ such that $\mathbf{a}(i) = 2A(i)-1$. Thus, $\mathbf{a}(i) = 1$ if $A(i) = 1$ and $\mathbf{a}(i) = -1$ if $A(i) = 0$. We can prove in $\mathbf{VTC}^0$ (by definition of

inner products and a product of a matrix and a vector—*innerprod* and *Matvecprod* function symbols, respectively, as defined in Section A.2.2) the following:

$$\mathbf{a}^t M\mathbf{a} = \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}\mathbf{a}(i)\mathbf{a}(j). \tag{A.42}$$

By assumption $\textsc{Mat}(M, \mathbf{C})$ holds (see Definition A.47) and so by definition A.47 and by (A.42) we can prove in $\mathbf{VTC}^0$ that:

$$\mathbf{a}^t M\mathbf{a} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=0}^{m-1} E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j), \tag{A.43}$$

where $E_{ij}^{(k)}$, for any $i, j \in [n]$, is:

$$E_{ij}^{(k)} := \begin{cases} +\frac{1}{2}, & x_i^{\varepsilon_i}, x_j^{\varepsilon_j} \in \mathbf{C}[k] \text{ and } \varepsilon_i \neq \varepsilon_j, \text{ for some } \varepsilon_i, \varepsilon_j \in \{0,1\} \text{ and } i \neq j; \\ -\frac{1}{2}, & x_i^{\varepsilon_i}, x_j^{\varepsilon_j} \in \mathbf{C}[k] \text{ and } \varepsilon_i = \varepsilon_j, \text{ for some } \varepsilon_i, \varepsilon_j \in \{0,1\} \text{ and } i \neq j; \\ 0, & \text{otherwise.} \end{cases} \tag{A.44}$$

By rearranging (A.43) we get

$$\mathbf{a}^t M\mathbf{a} = \sum_{k=0}^{m-1}\sum_{i=1}^{n}\sum_{j=1}^{n} E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j),$$

and since $E_{ij}^{(k)} = 0$ whenever either $x_i \notin \mathbf{C}[k]$ or $x_j \notin \mathbf{C}[k]$, we get

$$= \sum_{k=1}^{m-1}\sum_{i,j\in\{r\,:\,x_r\in\mathbf{C}[k]\}} E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j),$$

and further, since $E_{ij}^{(k)} = 0$ if $i = j$, and $E_{ij}^{(k)} = E_{ji}^{(k)}$, for any i, j, we have

$$= \sum_{k=0}^{m-1}\sum_{i<j\in\{r\,:\,x_r\in\mathbf{C}[k]\}} 2E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j). \tag{A.45}$$

Claim A.61. The theory $\mathbf{VTC}^0$ (in fact already $\mathbf{V}^0$) proves that for any $k = 0, \ldots, m-1$:

$$\sum_{i<j\in\{r\,:\,x_r\in\mathbf{C}[k]\}} 2E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j) = \begin{cases} +1, & \text{NAE}(\mathbf{C}[k], A); \\ -3, & \neg\text{NAE}(\mathbf{C}[k], A). \end{cases}$$

Proof of claim: For any $i < j \in \{r\,:\,x_r \in \mathbf{C}[k]\}$, if $A(i) \neq A(j)$ (which means that $\mathbf{a}(i) \neq \mathbf{a}(j)$) then $\mathbf{a}(i)\mathbf{a}(j) = -1$, and if $A(i) = A(j)$ (which means that $\mathbf{a}(i) = \mathbf{a}(j)$) then $\mathbf{a}(i)\mathbf{a}(j) = 1$. Thus, by (A.44), For any $i < j \in \{r\,:\,x_r \in \mathbf{C}[k]\}$:

$$E_{ij}^{(k)} = \begin{cases} +\frac{1}{2}, & \text{if } x_i^{\varepsilon_i} \neq x_j^{\varepsilon_j} \text{ under } \mathbf{a}; \\ -\frac{1}{2}, & \text{if } x_i^{\varepsilon_i} = x_j^{\varepsilon_j} \text{ under } \mathbf{a}. \end{cases} \tag{A.46}$$

Note that if $\mathrm{NAE}(\mathbf{C}[k], A)$ is true then there are exactly two pairs of literals $x_i^{\varepsilon_i}, x_j^{\varepsilon_j}$, $i < j$, for which $x_i^{\varepsilon_i}$ and $x_j^{\varepsilon_j}$ get different values under the assignment $\mathbf{a}$ (if A assigns 1 (i.e., $\top$) to one literal and 0 (i.e., $\bot$) to the other two literals, then two pairs have different values and one pair has the same value; and similarly if A assigns 0 to one literal and 1 to the other two literals). Therefore, if $\mathrm{NAE}(\mathbf{C}[k], A)$ is true then

$$\sum_{i<j\in\{r\,:\,x_r\in\mathbf{C}[k]\}} 2E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j) = 2\left(\frac{1}{2}+\frac{1}{2}-\frac{1}{2}\right) = 1.$$

On the other hand, if $\mathrm{NAE}(\mathbf{C}[k], A)$ is false then all pairs of literals $x_i^{\varepsilon_i}, x_j^{\varepsilon_j}$, $i < j$, get the same value under the assignment A, and so:

$$\sum_{i<j\in\{r\,:\,x_r\in\mathbf{C}[k]\}} 2E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j) = 2\left(-\frac{1}{2}-\frac{1}{2}-\frac{1}{2}\right) = -3.$$

■Claim

Let $Z = \{i < m : \mathrm{NAE}(\mathbf{C}[i], A)\}$ (note that $Z = \mathsf{satNAE}(A, \mathbf{C})$), and for any $k = 0, \ldots, m-1$, let $\gamma_k = \sum_{i<j\in\{r\,:\,x_r\in\mathbf{C}[k]\}} 2E_{ij}^{(k)}\mathbf{a}(i)\mathbf{a}(j)$. Then, by Claim A.61 and Proposition A.36:

$$\begin{aligned}\sum_{i=0}^{m-1}\gamma_i &= 1\cdot numones(Z) - 3\cdot(m - numones(Z))\\ &= 4\cdot numones(Z) - 3m\\ &= 4\cdot numones(\mathsf{satNAE}(A,\mathbf{C})) - 3m.\end{aligned} \tag{A.47}$$

By (A.45) we have

$$\sum_{i=0}^{m-1}\gamma_i = \mathbf{a}^t M\mathbf{a}, \tag{A.48}$$

and by the spectral inequality proved in Lemma A.68 in the next section, we have:

$$\mathbf{a}^t M\mathbf{a} \le \lambda n + o(1).$$

By (A.47) we thus get

$$4\cdot numones(\mathsf{satNAE}(A,\mathbf{C})) - 3m \le \lambda n + o(1),$$

which leads to

$$numones(\mathsf{satNAE}(A,\mathbf{C})) \le \frac{\lambda n + 3m}{4} + o(1).$$

□

A.5 The Spectral Bound

In this section we show how to prove inside $\mathbf{VTC}^0$ the desired spectral inequality, used in the proof of the key theorem (Theorem A.49; specifically, it was used in Lemma A.55 in Section A.4.2).

Since the original matrix associated to a 3CNF is a real symmetric matrix, and its eigenvectors and eigenvalues also might be real, and thus cannot be represented in our theory $\mathbf{VTC}^0$, we shall need to work with *rational approximations* of real numbers. We will work with polynomially small approximations. Specifically, a real number r in the real interval $[-1, 1]$ is represented with precision $1/n^c$, where n is the number of variables in the 3CNF and c is a constant natural number independent of n (that is, if $\widetilde{r}$ is the approximation of r, we shall have $|r - \widetilde{r}| \leq 1/n^c$). Recall that we will assume that all rational numbers have in fact the same denominator n^{2c} for some specific global constant c (see the Preliminaries, Section A.2.1 on this).

A.5.1 Notations

Here we collect the notation we use in this section. We denote by $e_1, \ldots, e_n$ the standard basis vectors spanning $\mathbb{Q}^n$. That is, for any $1 \leq i \leq n$ the vector $e_i \in \mathbb{Q}^n$ is 1 in the ith coordinate and all other coordinates are 0. For a vector $\mathbf{v}$ we denote by $\mathbf{v}(j)$ the jth entry in $\mathbf{v}$. Given a real symmetric matrix M we denote by $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^n$ the normalized eigenvectors of M. It is known that the collection of normalized eigenvectors of a symmetric $n \times n$ real matrix M forms an orthonormal basis for $\mathbb{R}^n$, called *the eigenvector basis of* M (cf. [53]). The (rational) approximation of the eigenvectors will be denoted $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{Q}^n$ and we define $v_{ij} := \mathbf{v}_i(j)$. Recall that for a real or rational vector $v = (v_1, \ldots, v_n)$ we denote by $\|v\|^2$ the squared Euclidean norm of v, that is, $\|v\|^2 = v_1^2 + \ldots + v_n^2$. We also define $\|\mathbf{v}\|_\infty := \max\{v_i : 1 \leq i \leq n\}$.

A.5.2 Rational approximations of Reals, vectors and matrices

Definition A.62 (Rational ε-approximation of a real number)**.** For $r \in \mathbb{R}$, we say that $q \in \mathbb{Q}$ is *a rational ε-approximation of r* (or just *ε-approximation*), if $|r - q| \leq \varepsilon$.

Claim A.63. For any real number $r \in [-1, 1]$ and any natural number m there exists a $1/m$-approximation of r whose numerator and denominator have values linearly bounded in m.

Proof of claim: By assumption, there exists an integer $0 \leq k < 2m$, such that $r \in \left[-1 + \frac{k}{m}, -1 + \frac{k+1}{m}\right]$. Then $-1 + \frac{k}{m}$ is a rational $1/m$-approximation of r. ■Claim

In a similar fashion we have:

Definition A.64 (Rational ε-approximation of (sets of) real vectors)**.** Let $0 < \varepsilon < 1$. For $\mathbf{u} \in \mathbb{R}^n$, we say that $\mathbf{v} \in \mathbb{Q}^n$ is *an ε-approximation of* $\mathbf{u}$, if $\mathbf{v}(i)$ is an ε-approximation of $\mathbf{u}(i)$, for all $i = 1, \ldots, n$. Accordingly, for a set $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\} \subseteq \mathbb{R}^n$, we say that $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subseteq \mathbb{Q}^n$ is a *(rational) ε-approximation of U* if every $\mathbf{v}_i \in \mathbb{Q}^n$ is an ε-approximation of the vector $\mathbf{u}_i$, $i = 1, \ldots, n$.

A.5.3 The predicate EigValBound

We define the predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$ which is meant to express the properties needed for the main proof. Basically, $\textsc{EigValBound}(M, \vec{\lambda}, V)$ expresses the fact that V is a rational $1/n^c$-approximation (Definition A.64) of the eigenvector basis of M, whose $1/n^c$-approximate eigenvalues (in decreasing order with respect to value) are $\vec{\lambda}$, for a sufficiently large constant $c \in \mathbb{N}$.

Note: For a number or a number term in the language, we sometimes use $|t|$ to denote the absolute value of t. This should not be confused with the length $|T|$ of a string term T.

Definition A.65 ($\textsc{EigValBound}$)**.** The predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$ is a Σ_0^B-definable relation in $\mathbf{VTC}^0$ that holds (in the standard two-sorted model) iff all the following properties hold (where $c \in \mathbb{N}$ is a sufficiently large global constant):

1. V is a sequence of n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{Q}^n$ with polynomially small entries. That is, for any $1 \leq i, j \leq n$, the rational number
$$v_{ij} := \mathbf{v}_i(j) \in \mathbb{Q}$$
is polynomial in n (meaning that both its denominator and numerator are polynomially bounded in n).

2. For any $1 \leq i, j \leq n$ it holds that the absolute value $|v_{ij}| \leq 2$.

3. For any $1 \leq i \leq n$, define:
$$\widetilde{e}_i := \sum_{j=1}^{n} v_{ij} \cdot \mathbf{v}_j \,.$$
Then, there exists $\mathbf{r}_i \in \mathbb{Q}^n$ for which
$$\widetilde{e}_i = e_i + \mathbf{r}_i \quad \text{and} \quad \|\mathbf{r}_i\|_\infty = O(1/n^{c-1}).$$
To formalize the existence of such an $\mathbf{r}_i$ we do not use an existential second-sort quantifier here; instead, we simply assert that for any $\ell = 1, \ldots, n$:
$$|\widetilde{e}_i(\ell) - e_i(\ell)| = O(1/n^{c-1}).$$

4. The vectors in V are "almost" orthonormal, in the following sense:
$$\begin{aligned} &\langle \mathbf{v}_i, \mathbf{v}_j \rangle = O(1/n^{c-1}) \,, && \text{for all } 1 \leq i \neq j \leq n, \\ &\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1 + O(1/n^{c-1}) \,, && \text{for all } 1 \leq i \leq n \,. \end{aligned}$$

5. The parameter $\vec{\lambda}$ is a sequence $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ of rational numbers such that for every $1 \leq i \leq n$, there exists a vector $\mathbf{t}_i \in \mathbb{Q}^n$ for which $\|\mathbf{t}_i\|_\infty = O(1/n^{c-3})$, and
$$M\mathbf{v}_i = \lambda_i \mathbf{v}_i + \mathbf{t}_i \,.$$
(Similar to Item 3 above, we do not use an existential second-sort quantifier for $\mathbf{t}_i$ here.)

It should be easy to check that $\textsc{EigValBound}(M, \vec{\lambda}, V)$ is a Σ_0^B-definable relation in $\mathbf{VTC}^0$.

Now we show that there exist objects $M, \vec{\lambda}, V$ that satisfy the predicate $\textsc{EigValBound}(M, \vec{\lambda}, V)$.

Proposition A.66 (Suitable approximations of eigenvector bases exist)**.** Let M be an $n \times n$ real symmetric matrix whose entries are quadratic in n. Let $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\} \subseteq \mathbb{R}^n$ be the orthonormal basis consisting of the eigenvectors of M, let $c \in \mathbb{N}$ be positive and constant (independent of n). If $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subseteq \mathbb{Q}^n$ is an $1/n^c$-approximation of U (Definition A.64), $\vec{\lambda} = \{\lambda_1, \ldots, \lambda_n\}$ is the collection of rational $1/n^c$-approximations of the real eigenvalues of M such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, then $\textsc{EigValBound}(M, \vec{\lambda}, V)$ holds (as before, the predicate holds in the standard two-sorted model, for the appropriate *encodings* of its parameters).[7]

Proof. Let u_{ij} be an abbreviation of $\mathbf{u}_i(j)$, that is, the jth element in the vector $\mathbf{u}_i$, and similarly for v_{ij}. We proceed by checking each of the conditions in Definition A.65.

Condition (1): Holds by the definition of an approximation of a real vector and by Claim A.63, stating that the ε-approximation of a real number in $[-1, 1]$ is a rational number whose both denominator and numerator are of value $O(n^c)$.

Condition (2): Since v_{ij} is a rational $1/n^c$-approximation of u_{ij}, and $|u_{ij}| \leq 1$ (because $\|\mathbf{u}_i\| = 1$) for any $1 \leq i, j \leq n$, we have $|v_{ij}| \leq 2$.

Condition (3): By orthonormality of the real matrix U, we have that $U^t = U^{-1}$, that is:

$$\sum_{i=1}^{n} u_{ij}\mathbf{u}_i = e_j \text{ , for any } j = 1, \ldots, n\,. \tag{A.49}$$

By assumption, for any $1 \leq i \leq n$ there exists $\mathbf{s}_i = (s_{i1}, \ldots, s_{in}) \in \mathbb{R}^n$ such that $\|\mathbf{s}_i\|_\infty \leq 1/n^c$ and $\mathbf{v}_i = \mathbf{u}_i + \mathbf{s}_i$. Therefore, for any $1 \leq j \leq n$, we have:

$$\begin{aligned}\widetilde{e}_j := \sum_{i=1}^{n} v_{ij}\mathbf{v}_i &= \sum_{i=1}^{n}(u_{ij} + s_{ij}) \cdot (\mathbf{u}_i + \mathbf{s}_i) \\ &= \underbrace{\sum_{i=1}^{n} u_{ij}\mathbf{u}_i}_{=e_j \text{ by (A.49)}} + \sum_{i=1}^{n} u_{ij}\mathbf{s}_i + \sum_{i=1}^{n} s_{ij} \cdot (\mathbf{u}_i + \mathbf{s}_i)\,. \end{aligned} \tag{A.50}$$

We define

$$\mathbf{r}_j := \sum_{i=1}^{n} u_{ij}\mathbf{s}_i + \sum_{i=1}^{n} s_{ij} \cdot (\mathbf{u}_i + \mathbf{s}_i)\,,$$

which gives us

$$\widetilde{e}_j = e_j + \mathbf{r}_j\,.$$

[7] This is an existence statement. We do not claim that the statement of the proposition is provable in the theory (nevertheless, some of the computations can be carried out inside the theory).

Note that since $\sum_{i=1}^{n} v_{ij}\mathbf{v}_i = \tilde{e}_j$ is a rational vector then $\mathbf{r}_j$ is also a rational vector.

It remains to show that $\|\mathbf{r}_j\|_\infty = O(1/n^{c-1})$. Since $1 = \|\mathbf{u}_i\|^2 = \sum_{j=1}^{n} u_{ij}^2$, we have $|u_{ij}| \leq 1$. By this, and by the fact that $\|\mathbf{s}_i\|_\infty \leq 1/n^c$, we get $\|\sum_{i=1}^{n} u_{ij}\mathbf{s}_i\|_\infty = O(1/n^{c-1})$, and $\|\sum_{i=1}^{n} s_{ij} \cdot (\mathbf{u}_i + \mathbf{s}_i)\|_\infty = O(1/n^{c-1})$. This means that $\|\mathbf{r}_j\|_\infty = O(1/n^{c-1})$.

Condition (4): This is similar to the proof of Condition (3). By assumption, for any $1 \leq i \leq n$ there exists $\mathbf{s}_i = (s_{i1}, \ldots, s_{in}) \in \mathbb{R}^n$ such that $\|\mathbf{s}_i\|_\infty \leq 1/n^c$, and $\mathbf{v}_i = \mathbf{u}_i + \mathbf{s}_i$. Thus, we have

$$\begin{aligned}\langle \mathbf{v}_i, \mathbf{v}_j \rangle &= \langle \mathbf{u}_i + \mathbf{s}_i, \mathbf{u}_j + \mathbf{s}_j \rangle \\ &= \langle \mathbf{u}_i, \mathbf{u}_j \rangle + \langle \mathbf{s}_i, \mathbf{u}_j + \mathbf{s}_j \rangle + \langle \mathbf{u}_i, \mathbf{s}_j \rangle \,. \end{aligned} \tag{A.51}$$

The first term in (A.51) is 0 since U is an orthonormal basis, and the second and third terms in (A.51) are both $O(1/n^{c-1})$ (by calculations similar to that in the proof of Condition (3)).

The proof of $\langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1 + O(1/n^{c-1})$, for all $1 \leq i \leq n$, is similar.

Condition (5): Similar to the proof of previous conditions, we define $\mathbf{s}_i = (s_{i1}, \ldots, s_{in}) \in \mathbb{R}^n$ such that $\|\mathbf{s}_i\|_\infty \leq 1/n^c$, and $\mathbf{v}_i = \mathbf{u}_i + \mathbf{s}_i$, for any $1 \leq i \leq n$. We have

$$\begin{aligned} M\mathbf{v}_i &= M(\mathbf{u}_i + \mathbf{s}_i) \\ &= M\mathbf{u}_i + M\mathbf{s}_i. \end{aligned} \tag{A.52}$$

Since $\mathbf{u}_i \in \mathbb{R}^n$ is the eigenvector of M and λ_i is a $1/n^c$-approximation of the eigenvalue of $\mathbf{u}_i$, we have that (A.52) equals

$$(\lambda_i + \epsilon)\mathbf{u}_i + M\mathbf{s}_i \tag{A.53}$$

for some $|\epsilon| \leq 1/n^c$,

$$\begin{aligned} &= \lambda_i\mathbf{u}_i + \epsilon\mathbf{u}_i + M\mathbf{s}_i \\ &= \lambda_i(\mathbf{v}_i - \mathbf{s}_i) + \epsilon\mathbf{u}_i + M\mathbf{s}_i \\ &= \lambda_i\mathbf{v}_i - \lambda_i\mathbf{s}_i + \epsilon\mathbf{u}_i + M\mathbf{s}_i \,. \end{aligned}$$

We put

$$\mathbf{t}_i := -\lambda_i\mathbf{s}_i + \epsilon\mathbf{u}_i + M\mathbf{s}_i.$$

It remains to show that $\|\mathbf{t}_i\|_\infty = O(1/n^{c-3})$.

Claim A.67. For every $1 \leq i \leq n$, $\lambda_i = O(n^3)$.

Proof of claim: Since $\|\mathbf{u}_i\|_\infty = 1$ and, by assumption, every entry in M is $O(n^2)$, we have:

$$\|M\mathbf{u}_i\|_\infty = O(n^3). \tag{A.54}$$

Observe that

$$M\mathbf{u}_i = (\lambda_i + \epsilon)\mathbf{u}_i = \lambda_i\mathbf{u}_i + \epsilon\mathbf{u}_i. \tag{A.55}$$

Because $|\epsilon| \leq 1/n^c$ and $\|\mathbf{u}_i\|_\infty = 1$, we have $\|\epsilon \mathbf{u}_i\|_\infty = O(1/n^c)$. Therefore, by (A.54) and (A.55) we have $\lambda_i = O(n^3)$. $\blacksquare_{\text{Claim}}$

We have $\|\mathbf{s}_i\|_\infty \leq 1/n^c$, and so by Claim A.67 we get that $\| - \lambda_i \mathbf{s}_i\|_\infty = O(1/n^{c-3})$. Now, $\|\epsilon \mathbf{u}_i\|_\infty = O(1/n^c)$ and since M has entries which are $O(n^2)$ we have $\|M\mathbf{s}_i\|_\infty = O(1/n^{c-3})$. We conclude that

$$\begin{aligned}\|\mathbf{t}_i\|_\infty &= \| - \lambda_i \mathbf{s}_i + \epsilon \mathbf{u}_i + M\mathbf{s}_i\|_\infty \\ &\leq \|-\lambda_i \mathbf{s}_i\|_\infty + \|\epsilon \mathbf{u}_i\|_\infty + \|M\mathbf{s}_i\|_\infty \\ &= O(1/n^{c-3}).\end{aligned}$$

□

A.5.4 Certifying the spectral inequality

In this section we show that the theory $\mathbf{VTC}^0$ can prove that,
if EigValBound$(M, \vec{\lambda}, V)$ holds, then the desired spectral inequality also holds.

Note on coding and formalizing the proof in $\mathbf{VTC}^0$: In what follows we will write freely terms such as matrices, vectors, inner products, products of a matrix by a vector (of the appropriate dimensions), addition of vectors, and big sums. We also use freely basic properties of these objects; like transitivity of inequalities, distributivity of a product over big sums, associativity of addition and product, etc. We showed how to formalize these objects, and how to prove their basic properties within $\mathbf{VTC}^0$ in Sections A.2.2 and A.2.2 (see Proposition A.39).

For an assignment $A \in \{0,1\}^n$ we define its associated vector $\mathbf{a} \in \{-1,1\}^n$ such that $\mathbf{a}(i) = 1$ if $A(i) = 1$ and $\mathbf{a}(i) = -1$ if $A(i) = 0$. In other words we define $\mathbf{a}(i) = 2A(i) - 1$. Note that

$$\mathbf{a} = \sum_{i=1}^{n} \mathbf{a}(i) \cdot e_i \,.$$

We define

$$\widetilde{\mathbf{a}} := \sum_{i=1}^{n} \mathbf{a}(i) \cdot \widetilde{e_i} \,, \tag{A.56}$$

and recall that $\widetilde{e_i} := \sum_{j=1}^{n} v_{ij} \cdot \mathbf{v}_j$ is a rational approximation of e_i (Definition A.65). We let $\mathbf{a}^t M \mathbf{a}$ abbreviate $\langle \mathbf{a}, M\mathbf{a} \rangle$ (which is Σ_1^B-definable in $\mathbf{VTC}^0$, by Section A.2.2).

Lemma A.68 (Main spectral bound)**.** *The theory $\mathbf{VTC}^0$ proves that if A is an assignment to n variables (that is, A is a string variable of length $n+1$) and* EigValBound$(M, \vec{\lambda}, V)$ *holds, then*

$$\mathbf{a}^t M \mathbf{a} \leq \lambda n + o(1) \,. \tag{A.57}$$

This is a corollary of Lemma A.69 and Lemma A.72 that follow.

Lemma A.69. *The theory* $\mathbf{VTC}^0$ *proves that for any assignment* A *to* n *variables,* EigValBound$(M, \vec{\lambda}, V)$ *implies:*

$$\mathbf{a}^t M \mathbf{a} \leq \widetilde{\mathbf{a}}^t M \widetilde{\mathbf{a}} + O(1/n^{c-5}),$$

where c *is the constant from the* EigValBound$(M_K, \vec{\lambda}, V)$ *predicate.*

Proof. First note that A is a string variable of length n. By Definition A.65 for any $1 \leq j \leq n$ there exists a vector $\mathbf{r}_j \in \mathbb{Q}^n$ such that $\widetilde{e}_j = e_j + \mathbf{r}_j$, and where $\|\mathbf{r}_j\|_\infty = O(1/n^{c-1})$. Therefore, by (A.56):

$$\widetilde{\mathbf{a}} = \sum_{i=1}^{n} \mathbf{a}(i)\widetilde{e}_i = \sum_{i=1}^{n} \mathbf{a}(i)(e_i + \mathbf{r}_i) = \sum_{i=1}^{n} \mathbf{a}(i)e_i + \sum_{i=1}^{n} \mathbf{a}(i)\mathbf{r}_i \,.$$

Note that $\sum_{i=1}^{n} \mathbf{a}(i)e_i = \mathbf{a}$, and let

$$\mathbf{r} := \sum_{i=1}^{n} \mathbf{a}(i)\mathbf{r}_i \,.$$

Then,

$$\widetilde{\mathbf{a}} = \mathbf{a} + \mathbf{r} \,,$$

and since $\mathbf{a}(i) \in \{-1, 1\}$, we have $\|\mathbf{r}\|_\infty = O(1/n^{c-2})$. Now, proceed as follows:

$$\begin{aligned} \mathbf{a}^t M \mathbf{a} &= (\widetilde{\mathbf{a}} - \mathbf{r})^t M (\widetilde{\mathbf{a}} - \mathbf{r}) \\ &= \widetilde{\mathbf{a}}^t M \widetilde{\mathbf{a}} - \widetilde{\mathbf{a}}^t M \mathbf{r} - \mathbf{r}^t M \widetilde{\mathbf{a}} + \mathbf{r}^t M \mathbf{r} \,. \end{aligned} \tag{A.58}$$

We now claim that (provably in $\mathbf{VTC}^0$) the three right terms in (A.58) are $o(1)$:

Claim A.70. The theory $\mathbf{VTC}^0$ proves that for any assignment A to n variables, EigValBound$(M, \vec{\lambda}, V)$ implies:

$$-\widetilde{\mathbf{a}}^t M \mathbf{r} - \mathbf{r}^t M \widetilde{\mathbf{a}} + \mathbf{r}^t M \mathbf{r} = O\left(1/n^{c-5}\right) \,.$$

Proof of claim: Consider $-\widetilde{\mathbf{a}}^t M \mathbf{r}$. Since $\|\widetilde{\mathbf{a}}\|_\infty \leq 2$ and since (by construction) each entry in M is at most $O(n^2)$, we have $\|\widetilde{\mathbf{a}}^t M\|_\infty = O(n^3)$. Therefore, since $\|\mathbf{r}\|_\infty \leq 1/n^{c-2}$, we get $-\widetilde{\mathbf{a}}^t M \mathbf{r} = O\left(\frac{1}{n^{c-5}}\right)$. Similarly, we have $-\mathbf{r}^t M \widetilde{\mathbf{a}} = O\left(\frac{1}{n^{c-5}}\right)$.

Considering $\mathbf{r}^t M \mathbf{r}$, we have $\|\mathbf{r}^t M\|_\infty = O(1/n^{c-4})$ and so $\mathbf{r}^t M \mathbf{r} = O(1/n^{c-5} \cdot 1/n^{c-2} \cdot n) = O(1/n^{2c-8}) = O(1/n^{c-5})$. ■Claim

Claim A.70 concludes the proof of Lemma A.69. □

Claim A.71. There is a constant c' such that the theory $\mathbf{VTC}^0$ proves that EigValBound$(M, \vec{\lambda}, V)$ implies that:

$$\begin{aligned} &\langle \widetilde{e}_i, \widetilde{e}_i \rangle = 1 + O(1/n^{c'}), && \text{for any } 1 \leq i \leq n, \text{ and} \\ &\langle \widetilde{e}_i, \widetilde{e}_j \rangle = O(1/n^{c'}), && \text{for any } 1 \leq i \neq j \leq n. \end{aligned}$$

Proof of claim: By assumption for any $1 \leq i \leq n$, $\widetilde{e}_i = e_i + \mathbf{r}_i$ for some $\|\mathbf{r}_i\|_\infty = O(1/n^{c-1})$. Thus

$$\begin{aligned}\langle \widetilde{e}_i, \widetilde{e}_i \rangle &= \langle e_i + \mathbf{r}_i, e_i + \mathbf{r}_i \rangle \\ &= \|e_i\|^2 + 2\langle e_i, \mathbf{r}_i \rangle + \|\mathbf{r}_i\|^2 \qquad \text{(A.59)} \\ &= 1 + o(1), \qquad \text{(A.60)}\end{aligned}$$

where the last equation holds since $2\langle e_i, \mathbf{r}_i \rangle$ and $\|\mathbf{r}_i\|^2$ can be easily proved to be $o(1)$ in $\mathbf{VTC}^0$.

Proving $\langle \widetilde{e}_i, \widetilde{e}_j \rangle = O(1/c')$ for any $1 \leq i \neq j \leq n$, is similar. $\blacksquare_{\text{Claim}}$

Lemma A.72. *The theory* $\mathbf{VTC}^0$ *proves that for any assignment* A *to* n *variables,* EigValBound$(M, \vec{\lambda}, V)$ *implies:*

$$\widetilde{\mathbf{a}}^t M \widetilde{\mathbf{a}} \leq \lambda n + o(1)\,. \qquad \text{(A.61)}$$

Proof. We have:

$$\begin{aligned}\widetilde{\mathbf{a}}^t M \widetilde{\mathbf{a}} &= \widetilde{\mathbf{a}}^t M \left(\sum_{i=1}^{n} \mathbf{a}(i) \widetilde{e}_i \right) && \text{(by definition of } \widetilde{\mathbf{a}}) \\ &= \widetilde{\mathbf{a}}^t M \left(\sum_{i=1}^{n} \left(\mathbf{a}(i) \cdot \sum_{j=1}^{n} v_{ji} \mathbf{v}_j \right) \right) && \text{(by definition of } \widetilde{e}_i) \\ &= \widetilde{\mathbf{a}}^t \sum_{i=1}^{n} \left(\mathbf{a}(i) \cdot \sum_{j=1}^{n} v_{ji} M \mathbf{v}_j \right) && \text{(rearranging)} \\ &= \widetilde{\mathbf{a}}^t \sum_{i=1}^{n} \left(\mathbf{a}(i) \cdot \sum_{j=1}^{n} v_{ji} (\lambda_j \mathbf{v}_j + \mathbf{r}_j) \right) && \text{(by Definition A.65)} \\ &= \widetilde{\mathbf{a}}^t \sum_{i=1}^{n} \left(\mathbf{a}(i) \cdot \sum_{j=1}^{n} \lambda_j v_{ji} \mathbf{v}_j \right) + \underbrace{\widetilde{\mathbf{a}}^t \sum_{i=1}^{n} \left(\mathbf{a}(i) \cdot \sum_{j=1}^{n} v_{ji} \mathbf{r}_j \right)}_{\text{Term 1}} && \text{(rearranging)}\end{aligned} \qquad \text{(A.62)}$$

We claim (inside $\mathbf{VTC}^0$) that the Term 1 above is of size $o(1)$:

Claim A.73. The theory $\mathbf{VTC}^0$ proves that for any assignment A to n variables, EigValBound$(M, \vec{\lambda}, V)$ implies

$$\widetilde{\mathbf{a}}^t \sum_{i=1}^{n} \left(\mathbf{a}(i) \cdot \sum_{j=1}^{n} v_{ji} \mathbf{r}_j \right) = O(1/n^{c-6})\,.$$

Proof of claim: The proof is similar to the proof of Claim A.70. Specifically, by Definition A.65, for any $1 \leq j \leq n$, we have $\|\mathbf{r}_j\|_\infty \leq 1/n^{c-1}$, and for any $1 \leq i, j \leq n$, we have $|v_{ji}| \leq 2$. Thus, $\mathbf{VTC}^0$ proves that $\|\sum_{j=1}^{n} v_{ji} \mathbf{r}_j\|_\infty = O(1/n^{c-2})$, for any $1 \leq i \leq n$. Since $\mathbf{a}(i) \in \{-1, 1\}$, for any $1 \leq i \leq n$, the

theory $\mathbf{VTC}^0$ proves $\|\mathbf{a}(i) \cdot \sum_{j=1}^n v_{ji}\mathbf{r}_j\|_\infty = O(1/n^{c-2})$, for any $1 \le i \le n$, and therefore also proves

$$\left\| \sum_{i=1}^n \left(\mathbf{a}(i) \cdot \sum_{j=1}^n v_{ji}\mathbf{r}_j \right) \right\|_\infty = O(1/n^{c-3}). \tag{A.63}$$

Now consider $\widetilde{\mathbf{a}} = \sum_{i=1}^n \mathbf{a}(i)\widetilde{e_i} = \sum_{i=1}^n \left(\mathbf{a}(i) \cdot \sum_{j=1}^n v_{ji}\mathbf{v}_j \right)$. Since, for any $1 \le i, j \le n$ we have $|v_{ji}| \le 2$ we have $\| \sum_{j=1}^n v_{ji}\mathbf{v}_j\|_\infty = O(n)$. Thus, since $\mathbf{a}(i) \in \{-1, 1\}$, $\mathbf{VTC}^0$ can prove that $\widetilde{\mathbf{a}} = O(n^2)$, and so by (A.63) the theory can finally prove

$$\widetilde{\mathbf{a}}^t \sum_{i=1}^n \left(\mathbf{a}(i) \cdot \sum_{j=1}^n v_{ij}\mathbf{r}_j \right) = O(1/n^{c-6}).$$

■Claim

It remains to bound the first term in (A.62):

$$\widetilde{\mathbf{a}}^t \cdot \left(\sum_{i=1}^n \mathbf{a}(i) \sum_{j=1}^n \lambda_j v_{ji}\mathbf{v}_j \right). \tag{A.64}$$

By the definition of $\widetilde{\mathbf{a}}$ in (A.56) and the definition of the $\widetilde{e_i}$'s, we get that (A.64) equals:

$$\left(\sum_{i=1}^n \mathbf{a}(i) \sum_{j=1}^n v_{ji}\mathbf{v}_j^t \right) \cdot \left(\sum_{i=1}^n \mathbf{a}(i) \sum_{j=1}^n \lambda_j v_{ji}\mathbf{v}_j \right). \tag{A.65}$$

We can prove in $\mathbf{VTC}^0$ that for any vectors $\mathbf{b}_1, \ldots, \mathbf{b}_\ell \in \mathbb{Q}^n$ and any rational numbers $c_1, \ldots, c_\ell$ and $\zeta_1, \ldots, \zeta_\ell$, such that $\zeta = \max\{\zeta_i : 1 \le i \le \ell\}$, we have

$$\left\langle \sum_{i=1}^\ell c_i\mathbf{b}_i, \sum_{i=1}^\ell \zeta_i c_i \mathbf{b}_i \right\rangle \le \zeta \cdot \left\langle \sum_{i=1}^\ell c_i\mathbf{b}_i, \sum_{i=1}^\ell c_i \mathbf{b}_i \right\rangle.$$

Therefore, we can prove in $\mathbf{VTC}^0$ that (A.65) is at most:

$$\begin{aligned}
&\lambda \cdot \left(\sum_{i=1}^n \mathbf{a}(i) \sum_{j=1}^n v_{ji}\mathbf{v}_j^t \right) \cdot \left(\sum_{i=1}^n \mathbf{a}(i) \sum_{j=1}^n v_{ji}\mathbf{v}_j \right) \\
&= \lambda \cdot \left(\sum_{i=1}^n \mathbf{a}(i)\widetilde{e_i}^t \right) \cdot \left(\sum_{i=1}^n \mathbf{a}(i)\widetilde{e_i} \right) && \text{(by definition of } \widetilde{e_i}) \\
&= \lambda \cdot \left\langle \sum_{i=1}^n \mathbf{a}(i)\tilde{e_i}\,,\, \sum_{i=1}^n \mathbf{a}(i)\tilde{e_i} \right\rangle \\
&= \lambda \cdot \sum_{i=1}^n \langle \mathbf{a}(i)\tilde{e_i}\,,\, \mathbf{a}(i)\tilde{e_i}\rangle + \lambda \cdot \sum_{1 \le i \ne j \le n} \langle \mathbf{a}(i)\tilde{e_i}, \mathbf{a}(i)\tilde{e_j}\rangle && \text{(by rearranging)} \\
&= \lambda \cdot \sum_{i=1}^n \mathbf{a}(i)^2 \langle \tilde{e_i}\,,\, \tilde{e_i}\rangle + \lambda \cdot \sum_{1 \le i \ne j \le n} \mathbf{a}(i)\mathbf{a}(j)\langle \tilde{e_i}, \tilde{e_j}\rangle && \text{(by rearranging again)} \\
&= \lambda \cdot \sum_{i=1}^n 1 \cdot (1 + o(1)) + \lambda \cdot \sum_{1 \le i \ne j \le n} \mathbf{a}(i)\mathbf{a}(j)o(1) && \text{(by Claim A.71)} \\
&= \lambda n + o(1) && \text{(for sufficiently large constant } c).^8
\end{aligned} \tag{A.66}$$

This concludes the proof of Lemma A.72. □

A.6 Wrapping-up the Proof: TC^0-Frege Refutations of Random 3CNF

In this section we establish the main result of this paper, namely, polynomial-size TC^0-Frege refutations for random 3CNF formulas with $\Omega(n^{1.4})$ clauses.

A.6.1 Converting the main formula into a $\forall\Sigma^B_0$ formula

Note that the main formula (Definition A.48) is a $\Sigma^B_0(\mathcal{L})$ formula, where the language $\mathcal{L}$ contains function symbols not in $\mathcal{L}^2_A$, and in particular it contains the *numones* function. Since Theorem A.42 relates $\mathbf{VTC}^0$ proofs of Σ^B_0 formulas to polynomial-size TC^0-Frege proofs, in order to use this theorem we need to convert the main formula into a Σ^B_0 formula (in the language $\mathcal{L}^2_A$). It suffices to show that $\mathbf{VTC}^0$ proves that the main formula is equivalent to a $\forall\Sigma^B_0$ formula, since if $\mathbf{VTC}^0$ proves a $\forall\Sigma^B_0$ formula $\forall\Phi$, it also proves the Σ^B_0 formula Φ obtained by discarding all the universal quantifiers in $\forall\Phi$.

Lemma A.74. *The theory $\mathbf{VTC}^0$ proves that the main formula is equivalent to a $\forall\Sigma^B_0$ formula $\forall\Phi$ where the universal quantifiers in the front of the formula all quantify over string variables that serve as counting sequences. Specifically,*

$$\forall\Phi := \forall Z_1 \leq t_1 \ldots \forall Z_r \leq t_r\, \Phi(Z_1, \ldots, Z_r), \tag{A.67}$$

where $t_1, \ldots, t_r$ are number terms and $\Phi(Z_1, \ldots, Z_r)$ has also free variables other then the Z_i's, and every occurrence of every Z_i appears in Φ in the form $(\delta_{\mathsf{NUM}}(|T|, T, Z_i) \wedge Z_i[t] = s)$, for some string term T and number terms t, s, and where $\delta_{\mathsf{NUM}}(|T|, T, Z_i)$ states that Z_i is a counting sequence *that counts the number of ones in T until position $|T|$ (see Definition A.29).*

Proof. The following steps convert the main formula into a $\forall\Sigma^B_0$ formula which is equivalent (provably in $\mathbf{VTC}^0$) to the main formula:

1. All the functions in the main formula are $\mathbf{AC}^0$-reducible to $\mathcal{L}^2_A \cup \{numones\}$ (see Section A.2.2). Thus, the defining axioms of all the function symbols in the main formula can be assumed to be $\Sigma^B_0(numones)$ formulas. Now, it is a standard procedure to substitute in the main formula all the function symbols by their $\Sigma^B_0(numones)$-defining axioms.[9] The resulting formula is $\Sigma^B_0(numones)$, and provably in $\mathbf{VTC}^0$ is equivalent to the original main formula.

[8] The constant c here is the global constant power of n (appearing in the $1/n^c$-approximation in Definition A.65).

[9] When the defining axiom of a string function $F(\vec{x}, \vec{X})$ is a *bit-definition* $i < r(\vec{x}, \vec{X}) \wedge \psi(i, \vec{x}, \vec{X})$, we substitute an atomic formula like $F(\vec{x}, \vec{X})(z)$, by $z < r(\vec{x}, \vec{X}) \wedge \psi(z, \vec{x}, \vec{X})$ (cf. Lemma V.4.15 in [29]).

2. We now substitute all the *numones* function symbols by their Σ_1^B-defining axioms. Specifically, every occurrence of $numones(t, T)$ in the formula, for t, T number and string terms, respectively, occurs inside some atomic formula $\Psi := \Psi(\dots numones(t, T) \dots)$. And so we substitute Ψ by the existential formula

$$\exists Z \leq 1 + \langle |T|, |T| \rangle (\delta_{\mathsf{NUM}}(|T|, T, Z) \wedge Z[t] = z \wedge \Psi(\dots z \dots)).$$

3. Note that all the *numones* function symbols appear in the *premise* of the implication in the main formula, so we can take all these existential quantifiers out of the premise of the implication and obtain a universally quantified formula, where the universal quantifiers in the front of the formula all quantify over string variables that serve as counting sequences (as in Item 2 above).

□

A.6.2 Propositional proofs

We need to restate the main probabilistic theorem in [37]:

Theorem A.75 ([37], Theorem 3.1)**.** *Let* $\mathbf{C}$ *be a random 3CNF with* n *variables and* $m = \beta \cdot n$ *clauses (*$\beta = c \cdot n^{0.4}$*,* c *some fixed large constant). Then, with probability converging to* 1*, the following holds:*

- *The imbalance of* $\mathbf{C}$ *is at most* $O(n\sqrt{\beta}) = O(n^{1.2})$*.*
- *The largest eigenvalue* λ *satisfies* $\lambda = O(\sqrt{\beta}) = O(n^{0.2})$*.*
- *There are* $k = O(\frac{n}{\beta^2}) = O(n^{0.2})$*,* $t = \Omega(n\beta) = \Omega(n^{1.4})$*,* $d = O(k) = O(n^{0.2})$ *and* $\mathcal{C}$ *with* $|\mathcal{C}| = t$ *such that* $\textsc{Coll}(t, k, d, n, m, \mathbf{C}, \mathcal{C})$ *holds.*

We need to rephrase the theorem in a manner that suites our needs, as follows:

Corollary A.76. Let $\mathbf{C}$ be random 3CNF with n variables and $m = c \cdot n^{1.4}$ clauses where c is sufficiently large constant. Then, with probability converging to 1, the following holds:[10]

1. There exists an $I = O(n^{1.2})$ such that $\textsc{Imb}(\mathbf{C}, I)$.
2. There exists an $1/n^{c'}$-rational approximation V of the eigenvector matrix of M and $1/n^{c'}$-rational approximations $\vec{\lambda}$ of the eigenvalues of M, for some constant $c' > 6$; in other words, $\textsc{EigValBound}(M, \vec{\lambda}, V)$ and $\textsc{Mat}(M, \mathbf{C})$ hold. And the $1/n^{c'}$-rational approximation λ of the largest eigenvalue of M satisfies $\lambda = O(n^{0.2})$.

[10] Formally speaking, we mean that the following three items hold in the standard two-sorted model $\mathbb{N}_2$, when all the second-sort objects (like $\mathbf{C}$ and $\mathscr{D}$) are in fact finite sets of numbers (encoding $\mathbf{C}$ and $\mathscr{D}$), natural numbers are treated as natural numbers in the standard two-sorted model and rational numbers are the corresponding natural numbers that encode them as pairs of natural numbers (as described in Section A.2.1).

3. There are natural numbers $k = O(n^{0.2})$, $t = \Omega(n^{1.4})$, $d = O(k) = O(n^{0.2})$ and a sequence $\mathscr{D}$ of t inconsistent k-tuples such that $\textsc{Coll}(t, k, d, n, m, \mathbf{C}, \mathscr{D})$ holds, and such that:
$$t > \frac{d(I + \lambda n)}{2} + o(1)\,.$$

Proof. The corollary stems directly from Theorem A.75. Note only that the last inequality concerning t stems from direct computations, using the bounds in Theorem A.75 with $\beta = n^{0.4}$, and that Item 2 follows from Proposition A.66. □

Recall the premise in the implication in the main formula:

$$\begin{aligned}\text{3CNF}(\mathbf{C}, n, m) \wedge \textsc{Coll}(t, k, d, n, m, \mathbf{C}, \mathscr{D}) \wedge \textsc{Imb}(\mathbf{C}, I) \wedge \textsc{Mat}(M, \mathbf{C}) \wedge \\ \textsc{EigValBound}(M, \vec{\lambda}, V) \wedge \lambda = \max\{\vec{\lambda}\} \wedge t > \frac{d \cdot (I + \lambda n)}{2} + o(1).\end{aligned} \tag{A.68}$$

Let $\text{PREM}(\mathbf{C}, n, m, t, k, d, \mathscr{D}, I, \vec{\lambda}, V, M, \lambda, \vec{Z})$ be the formula obtained from (A.68) after transforming the main formula into a $\forall\Sigma^B_0$ formula, where $\vec{Z}$ is a sequence of strings variables for counting sequences added after the transformation (as described in Lemma A.74).

The following is a simple claim about the propositional translation (given without a proof):

Claim A.77. If a Σ^B_0 formula $\varphi(\vec{x}, \vec{X})$ can be evaluated to a true sentence in $\mathbb{N}_2$ by assigning numbers $\underline{\vec{x}}$ and sets $\underline{\vec{X}}$ to the appropriate variables, then the translation $[\![\varphi]\!]_{\underline{\vec{x}}, |\underline{\vec{X}}|}$ is satisfiable.

Lemma A.78. *For every $m, n \in \mathbb{N}$ and every unsatisfiable 3CNF formula $\mathbf{C}$ with m clauses and n variables such that $\text{PREM}(\mathbf{C}, n, m, \dots)$ is true for some assignment to the remaining variables (i.e. to the unspecified variables denoted by "…"; this also implies that $[\![\text{PREM}(\mathbf{C}, n, m, \dots)]\!]$ is satisfiable), there exists a polynomially bounded TC^0-Frege proof of $\neg\mathbf{C}$ (i.e. the sequent $\longrightarrow \neg\mathbf{C}$ can be derived).*

Proof. Recall that for given $m, n \in \mathbb{N}$, 3CNF formula $\mathbf{C} = (\mathbf{C}[\alpha])_{\alpha<m}$ and assignment A, the formula $\exists\alpha \leq m\, \textsc{NotSAT}(\mathbf{C}[i], A)$ (which is the consequence of the implication in the main formula A.48) is the statement:

$$\begin{aligned}\exists\alpha < m \exists i, j, k \leq n \big(\quad & \langle \mathbf{C}[\alpha] \rangle^5_1 = i \wedge (A(i) \leftrightarrow \langle \langle \mathbf{C}[\alpha] \rangle^5_4 \rangle^3_1 = 0) \\ & \wedge \langle \mathbf{C}[\alpha] \rangle^5_2 = j \wedge (A(j) \leftrightarrow \langle \langle \mathbf{C}[\alpha] \rangle^5_4 \rangle^3_2 = 0) \\ & \wedge \langle \mathbf{C}[\alpha] \rangle^5_3 = k \wedge (A(k) \leftrightarrow \langle \langle \mathbf{C}[\alpha] \rangle^5_4 \rangle^3_3 = 0) \big).\end{aligned}$$

The propositional translation of this formula (Definition A.40) contains the variables $p^{\mathbf{C}}_{\langle i,j,k,\ell,\alpha \rangle}$ with $i, j, k \leq n$, $\alpha < m$. Additionally it contains variables p^A_i for $i \leq n$ stemming from the assignment A. It is not necessary to show the full translation of the formula, since we intend to plug-in propositional constants $(\top, \bot)$ for some of the variables. In other words, parts of the formula will consist

of only constants and so it is unnecessary to give these parts in full detail. Having this in mind, the translation $[\![\exists\alpha < m\text{NOTSAT}(\mathbf{C}[\alpha], A)]\!]_{m,n}$ is

$$\begin{aligned}\bigvee_{\alpha=0}^{m-1}\bigvee_{i,j,k=1}^{n}\big(\quad & ([\![\langle\mathbf{C}[\alpha]\rangle_1^5 = i]\!]_{m,n} \wedge (p_i^A \leftrightarrow [\![\langle\langle\mathbf{C}[\alpha]\rangle_4^5\rangle_1^3 = 0]\!]_{m,n})) \\ & \wedge (\langle[\![\langle\mathbf{C}[\alpha]\rangle_2^5 = j]\!]_{m,n} \wedge (p_j^A \leftrightarrow [\![\langle\langle\mathbf{C}[\alpha]\rangle_4^5\rangle_2^3 = 0]\!]_{m,n})) \\ & \wedge (\langle[\![\langle\mathbf{C}[\alpha]\rangle_3^5 = k]\!]_{m,n} \wedge (p_k^A \leftrightarrow [\![\langle\langle\mathbf{C}[\alpha]\rangle_4^5\rangle_3^3 = 0]\!]_{m,n}))).\end{aligned} \tag{A.69}$$

Here, the variables $p^{\mathbf{C}}_{\langle i,j,k,\ell,\alpha\rangle}$ all implicitly appear in the parts inside $[\![\cdot]\!]$.

Now assume we have a fixed 3CNF $\underline{\mathbf{C}}$ with n variables and m clauses. Then for every $\alpha < m$ there exists $1 \leq i, j, k \leq n$ such that the formulas $[\![\langle\underline{\mathbf{C}[\alpha]}\rangle_1^5 = i]\!]_{m,n}$ and $[\![\langle\underline{\mathbf{C}[\alpha]}\rangle_2^5 = j]\!]_{m,n}$ and $[\![\langle\underline{\mathbf{C}[\alpha]}\rangle_3^5 = k]\!]_{m,n}$ are all satisfied (in fact they are polynomial-size in n propositional tautologies consisting of only constants $\top, \bot$). From now on we will only concentrate on the disjuncts where this is the case (as the other disjuncts are falsified, or in other words they are propositional contradictions consisting of only constants).

By plugging $\underline{\mathbf{C}}$ into $[\![\langle\langle\mathbf{C}[\alpha]\rangle_4^5\rangle_1^3 = 0]\!]_{m,n}$ and $[\![\langle\langle\mathbf{C}[\alpha]\rangle_4^5\rangle_2^3 = 0]\!]_{m,n}$ and $[\![\langle\langle\mathbf{C}[\alpha]\rangle_4^5\rangle_3^3 = 0]\!]_{m,n}$ we get that $[\![\exists\alpha < m\text{NOTSAT}(\mathbf{C}[\alpha], A)]\!]_{m,n}$ is evaluated to

$$\bigvee_{\alpha<m}\left((p_i^A)^{\ell_1^\alpha} \wedge (p_j^A)^{\ell_2^\alpha} \wedge (p_k^A)^{\ell_3^\alpha}\right), \tag{A.70}$$

where ℓ_r^α is an abbreviation of $[\![\langle\langle\underline{\mathbf{C}[\alpha]}\rangle_4^5\rangle_r^3 = 0]\!]_{m,n}$, and thus we can observe that (A.70) gets evaluated to $\neg\underline{\mathbf{C}}(p_1^A/x_1, \ldots, p_n^A/x_n)$, where p_i^A/x_i means substitution of x_i by p_i^A.

By Theorem A.49 the theory $\mathbf{VTC}^0$ proves the main formula and so by Lemma A.74 there is a $\mathbf{VTC}^0$ proof of

$$\text{PREM}(\mathbf{C}, n, m, t, k, d, \mathscr{D}, I, \vec{\lambda}, V, M, \lambda, \vec{Z}) \rightarrow \exists i < m\,\text{NOTSAT}(\mathbf{C}[i], A).$$

Thus, by Theorem A.42 we can derive a polynomially bounded TC^0-proof of the formula

$$[\![\text{PREM}(\mathbf{C}, \ldots)]\!]_{m,n} \rightarrow [\![\exists\alpha < m\text{NOTSAT}(\mathbf{C}[\alpha], A)]\!]_{m,n}$$

and thus also of the *sequent*

$$[\![\text{PREM}(\mathbf{C}, \ldots)]\!]_{m,n} \longrightarrow [\![\exists\alpha < m\text{NOTSAT}(\mathbf{C}[\alpha], A)]\!]_{m,n}.$$

By Claim A.77 and the assumption that $\text{PREM}(\underline{\mathbf{C}}, n, m, \ldots)$ is true in $\mathbb{N}_2$ for an assignment to the remaining variables we know that $[\![\text{PREM}(\underline{\mathbf{C}}, \ldots)]\!]_{m,n}$ is satisfiable. Plugging-in such a satisfying assignment $\vec{a}$ into $[\![\text{PREM}(\underline{\mathbf{C}}, \ldots)]\!]_{m,n}$, Lemma A.17 yields a polynomially bounded TC^0-Frege proof of

$$[\![\text{PREM}(\underline{\mathbf{C}}, \vec{a})]\!]_{m,n}$$

and of the sequent

$$[\![\text{PREM}(\underline{\mathbf{C}}, \vec{a})]\!]_{m,n} \longrightarrow [\![\exists\alpha < m\text{NOTSAT}(\underline{\mathbf{C}[\alpha]}, A)]\!]_{m,n}.$$

Using the Cut rule (Definition A.14) we get a polynomially bounded TC^0-Frege proof of the formula

$$[\![\exists \alpha < m \text{NOTSAT}(\underline{\mathbf{C}}[\alpha], A)]\!]_{m,n}.$$

As we showed before, this gets evaluated to

$$\neg \underline{\mathbf{C}}(p_1^A/x_1, \ldots, p_n^A/x_n)$$

as desired. Because of Claim A.17, this proof is only polynomially longer than the one of the translation of the main formula. Since that proof was polynomially bounded, the above proof of $\neg \underline{\mathbf{C}}(p_i^A/x_i)$ also is. □

We can now conclude:

Corollary A.79. With probability converging to 1, a random 3CNF $\mathbf{C}$ with n variables and $m \geq c \cdot n^{1.4}$ clauses, c a sufficiently large constant, $\neg\mathbf{C}$ has polynomially bounded TC^0-Frege proofs, while $\mathbf{C}$ has no sub-exponential size resolution refutations (as long as $m = O(n^{1.5-\epsilon})$, for $0 < \epsilon < 1/2$).

Proof. By Corollary A.76, with probability converging to 1 there exists an assignment of numbers and strings $\vec{\alpha}$ (including also the appropriate counting sequences assigned to the Z_i string variables introduced in Lemma A.74) such that $\text{PREM}(\mathbf{C}, \vec{\alpha})$ holds (in the standard two-sorted model). Therefore, with probability converging to 1 we can apply Lemma A.78 to establish that $\neg\mathbf{C}$ has a short TC^0-Frege proof. That with probability converging to 1 there are no sub-exponential size resolution refutations of $\mathbf{C}$ follows from [25, 10, 15]. □

A.7 Acknowledgments

We wish to thank Jan Krajíček for very helpful discussions concerning the topic of this paper and for commenting on an earlier manuscript and Emil Jeřabek and Neil Thapen for answering many of our questions about theories of weak arithmetic and for many other insightful comments.

B. Polylogarithmic Cuts of Models of $\mathbf{V}^0$

B.1 Introduction

This article is on the one hand on models of weak arithmetics and on the other on proof complexity, i.e. the question of how long formal proofs of tautologies have to be in given proof systems. Therefore the introduction will consist of two parts, one for each subject.

Models of weak arithmetics, like $\mathbf{I\Delta}_0$, have been extensively studied for several reasons, possibly being the simplest objects whose theories bear enough strength to do a good part of mathematics in and still are weak enough to allow for a certain kind of constructiveness. The latter has been demonstrated over and over again by various results connecting weak arithmetic theories with complexity classes and computability. We are interested in the strength of the theory obtained by restricting our objects of reasoning to a small initial part of a given model. Since a two-sorted theory, such as $\mathbf{V}^0$, is much stronger on its number part than on its set part, it is likely that such a cut is a model of a supposedly much stronger theory. Indeed we will see in Section B.3 that certain cuts of models of $\mathbf{V}^0$ are models of the provably stronger theory $\mathbf{VNC}^1$. This strengthens a result by Paris and Wilkie [75][74], who show that such cuts are models of $\mathbf{VTC}^0$. In fact they work in a more general setting and, following our argumentation, their result readily implies the sub exponential simulation of TC^0-Frege by AC^0-Frege from Bonet, Domingo, Gavaldà, Maciel, and Pitassi [17].

Proof Complexity, on the other hand, more or less began when Cook and Reckhow [30] discovered the close connection between the lengths of formal proofs in propositional proof systems and standard complexity classes. This connection yields a possibility of dealing with the $\mathsf{coNP/NP}$ question by asking, whether there exists a propositional proof system that is polynomially bounded. We will not directly address this question here, but rather explore the relative strengths of two major proof systems, Frege and bounded depth Frege. These proof systems have been extensively studied, due to their natural appearance as classical calculi, such as Gentzen's PK, and it is well known that Frege systems are stronger than bounded depth Frege systems, as the former system has polynomial size proofs for the Pigeonhole Principle (see [21]), while the latter does not (see [62] and [78]). Lately, Filmus, Pitassi and Santhanam [39] have proved a sub exponential simulation of Frege by bounded depth Frege using a combinatoric argument. In Section B.4 we will obtain the same result by an application of our result about cuts to the provability of the Reflection Principle for Frege in bounded depth Frege. Currently Cook, Ghasemloo and Nguyen [28] are working on a purely syntactical proof that gives a slightly better result with respect to the strength of the simulated proof system.

The paper is built-up as follows. In section B.2 we briefly recapture some basics about Complexity Theory, Bounded Arithmetic, Proof Complexity and the various connections between them. As this is only expository it might be helpful to consult some of the references for a more detailed introduction (see [5], [29]

and [57]). After that, in Section B.3 we prove a formalization of Nepomnjascij's Theorem in the polylogarithmic cut of a model of $\mathbf{V}^0$. Using a standard algorithm for evaluating circuits and then applying the formalized version of Nepomnjascij's Theorem we can conclude that this cut is indeed a model of $\mathbf{VNC}^1$. Finally, in Section B.4, we apply this result to prove that a version of the Bounded Reflection Principle of Frege is provable in $\mathbf{V}^0$. This, together with a standard argument linking the provability of Reflection Principles with simulation results, yields the sub exponential simulation of Frege by bounded depth Frege.

B.2 Preliminaries

We assume familiarity with Turing machines, circuits and standard complexity classes such as P, NP, $\mathsf{TimeSpace}(f, g)$, NC^i, AC^i and so on. See for example [5] for an introduction. We will not work a lot within these classes, but rather apply known relations between such classes and weak arithmetic theories.

We will work in a two-sorted arithmetic setting, having one sort of variables representing numbers and the second sort representing bounded sets of numbers. We identify such bounded subsets with strings. See [29] for a thorough introduction. The underlying language, denoted $\mathcal{L}_A^2$, consists of the following relation, function and constant symbols:

$$\{+, \cdot, \leq, 0, 1, |\cdot|, =_1, =_2, \in\} .$$

An $\mathcal{L}_A^2$-structure M consists of a first-sort universe U_1^M of numbers and a second-sort universe U_2^M of bounded subsets of numbers. If M is a model of the two-sorted theory $\mathbf{V}^0$ (see B.2.2), then the functions $+$ and $\cdot$ are the addition and multiplication on the universe of numbers. 0 and 1 are interpreted as the appropriate elements zero and one with respect to addition and multiplication. The relation $\leq$ is an ordering relation on the first-sort universe. The function $|\cdot|$ maps an element of the set sort to its largest element plus one (i.e. to an element of the number sort). The relation $=_1$ is interpreted as equality between numbers, $=_2$ is interpreted as equality between bounded sets of numbers. The relation $\in$ holds for a number n and a set of numbers N if and only if n is an element of N. The standard model of two-sorted Peano Arithmetic will be denoted as $\mathbb{N}_2$. It consists of a first-sort universe $U_1 = \mathbb{N}$ and a second-sort universe U_2 of all finite subsets of $\mathbb{N}$. The symbols are interpreted in the usual way.

We denote the first-sort (number) variables by lower-case letters $x, y, z, ...$, and the second-sort (set) variables by capital letters $X, Y, Z, ...$ We can build formulas in the usual way, using two sorts of quantifiers, number quantifiers and string quantifiers. A number quantifier $\exists x$ $(\forall x)$ is bounded if it is of the form $\exists x(x \leq f \wedge \dots)$ $(\forall x(x \leq f \rightarrow \dots))$ for some number term f. A string quantifier $\exists X$ $(\forall X)$ is bounded if it is of the form $\exists X(|X| \leq f \wedge \dots)$ $(\forall X(|X| \leq f \rightarrow \dots))$ for some number term f. A formula is bounded iff all its quantifiers are. All formulas in this paper will be bounded. A formula φ is in Σ_0^B (or Π_0^B) if it uses no string quantifiers and all number quantifiers are bounded. A formula φ is a Σ_{i+1}^B (or Π_{i+1}^B) if it is of the form $\exists X_1 \leq p(n) \dots \exists X_m \leq p(n)\psi$ (or $\forall X_1 \leq p(n) \dots \forall X_m \leq p(n)\psi$), where $\psi \in \Pi_i^B$ (or $\psi \in \Sigma_i^B$, respectively). If a relation or predicate can be defined by both a Σ_i^B and a Π_i^B formula, then we call it Δ_i^B

definable. The *depth* of a formula is the maximal number of alternations of its logical connectives and quantifiers.

As mentioned before we will represent a bounded set of numbers N by a finite string $S_N = S_N^0 \dots S_N^{|N|-1}$ such that $S_N^i = 1$ if and only if $i \in N$. We will abuse notation and identify bounded sets and strings, i.e. N and S_N.

Further, we will encode monotone propositional formulas inductively as binary trees in the standard way, giving a node the value 1 if it corresponds to a conjunction and the value 0, if it corresponds to a disjunction. Binary trees are encoded as strings as follows. If position x contains the value of a node $\mathfrak{n}_x$, then the value of its left successor is contained in position $2x$, while the value of its right successor is in $2x + 1$.

B.2.1 Elements of Proof Complexity

We restate some basic definitions introduced in [30].

Definition B.1. A *propositional proof system (pps)* is a surjective polynomial-time function $P : \{0,1\}^* \longrightarrow \mathsf{TAUT}$, where TAUT is the set of propositional tautologies (in some natural encoding). A string π with $P(\pi) = \tau$ is called a *P-proof* of τ.

We can define a quasi ordering on the class of all pps as follows.

Definition B.2. Let P, Q be propositional proof systems.

- P simulates Q (in symbols $P \geq Q$), iff there is a polynomial p, such that for all $\tau \in \mathsf{TAUT}$ there is a π_P with $P(\pi_P) = \tau$, such that for all π_Q with $Q(\pi_Q) = \tau$, $|\pi_P| \leq p(|\pi_Q|)$.
- If there is a polynomial time machine that takes Q-proofs and produces P-proofs for the same formula we say that P p-simulates Q (in symbols $P \geq_p Q$).
- If P and Q mutually (p-)simulate each other, we say that they are (p-)equivalent (in symbols $P \equiv Q$ and $P \equiv_p Q$, respectively).

In this article we will be mainly interested in bounded depth Frege systems and some of their extensions. A Frege system is a typical textbook proof system, such as Gentzen's propositional calculus PK. We will only sketch a single rule of such a system as an example and refer the interested reader to standard logic textbooks.

$$\frac{\Gamma \longrightarrow A, \Delta \qquad \Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}\,(\text{Cut})$$

Here, Δ and Γ are sets of formulas while A is a formula. $\Gamma \longrightarrow \Delta$ is read as "The conjunction of all formulas in Γ implies the disjunction of all formulas in Δ". The Cut Rule therefore says that, if Γ implies A or Δ, and Γ and A imply Δ, then Γ already implies Δ. The formula A is called the *Cut Formula*.

In a bounded depth Frege system the depths of all formulas in a derivation are bounded by some global constant. This is equivalent to being representable by an AC^0 circuit. Thus we also call bounded depth Frege AC^0-Frege. If the formulas are unbounded, we speak of NC^1-Frege or simply of Frege. We readily get

Fact 2. AC^0-Frege $\leq_p$ Frege.

A pps P is *polynomially bounded* iff there is a polynomial p such that every tautology τ has a P-proof π with $|\pi| \leq p(|\tau|)$.

We are interested in the existence of polynomially bounded pps. This is, at least in part, due to the following theorem.

Fact 3 ([30])**.** $\mathsf{NP} = \mathsf{coNP} \Leftrightarrow$ There exists a polynomially bounded pps.

An easier task than searching for a polynomially bounded pps might be to find some pps with sub exponential bounds to the lengths of proofs. This corresponds to the question, whether sub exponential time nondeterministic Turing machines can compute coNP-complete languages. To explore the existence of such systems we generalize Definition B.2.

Definition B.3. Let P, Q be propositional proof systems and F a family of increasing functions on $\mathbb{N}$.

- P F-simulates Q (in symbols $P \geq^F Q$), iff there is a function $f \in F$, such that for all $\tau \in \mathsf{TAUT}$ there is a π_P with $P(\pi_P) = \tau$, such that for all π_Q with $Q(\pi_Q) = \tau$, $|\pi_P| \leq f(|\pi_Q|)$.
- If there is an $\mathsf{Time}(F)$-machine that that takes Q-proofs and produces P-proofs for the same formula we say that we say that P F-computably simulates Q (in symbols $P \geq_p^F Q$).
- If P and Q mutually F-(computably) simulate each other, we say that they are F-(computably) equivalent (in symbols $P \equiv^F Q$ and $P \equiv_p^F Q$, respectively).

We say a pps P *sub exponentially simulates* a pps Q iff the above F can be chosen as a class of $2^{n^{o(1)}}$ functions.

B.2.2 The theory $\mathbf{V}^0$

The base theory we will be working with is $\mathbf{V}^0$. It consists of the following axioms:

Basic 1. $x + 1 \neq 0$	**Basic 2**. $x + 1 = y + 1 \rightarrow x = y$
Basic 3. $x + 0 = x$	**Basic 4**. $x + (y + 1) = (x + y) + 1$
Basic 5. $x \cdot 0 = 0$	**Basic 6**. $x \cdot (y + 1) = (x \cdot y) + x$
Basic 7. $(x \leq y \wedge y \leq x) \rightarrow x = y$	**Basic 8**. $x \leq x + y$
Basic 9. $0 \leq x$	**Basic 10**. $x \leq y \vee y \leq x$
Basic 11. $x \leq y \leftrightarrow x < y + 1$	**Basic 12**. $x \neq 0 \rightarrow \exists y \leq x(y + 1 = x)$
L1. $X(y) \rightarrow y < \lvert X\rvert$	**L2**. $y + 1 = \lvert X\rvert \rightarrow X(y)$

SE. $(\lvert X\rvert = \lvert Y\rvert \wedge \forall i \leq \lvert X\rvert\,(X(i) \leftrightarrow Y(i))) \rightarrow X = Y$

Σ_0^B**-COMP.** $\exists X \leq y \forall z < y(X(z) \leftrightarrow \varphi(z))\,,$ for all $\varphi \in \Sigma_0^B$.

Here, the Axioms **Basic 1** through **Basic 12** are the usual axioms used to define Peano Arithmetic without induction (PA^-), which settle the basic properties of Addition, Multiplication, Ordering, and of the constants 0 and 1. The Axiom **L1** says that the length of a string coding a finite set is an upper bound to the size of its elements. **L2** says that $|X|$ gives the largest element of X plus 1. **SE** is the extensionality axiom for strings which states that two strings are equal if they code the same sets. Finally, Σ_0^B**-COMP** is the comprehension axiom schema for Σ_0^B-formulas (it is an axiom for each such formula) and implies the existence of all sets, which contain exactly the elements that fulfill any given Σ_0^B property.

Fact 4. The theory $\mathbf{V}^0$ proves the Induction Axiom schema for Σ_0^B formulas Φ:

$$(\Phi(0) \wedge \forall x(\Phi(x) \to \Phi(x+1))) \to \forall z\, \Phi(z).$$

When speaking about theories we will always assume that the theories are two-sorted theories as in [29]).

The following is a basic notion:

Definition B.4 (Two-sorted definability)**.** Let $\mathcal{T}$ be a theory over the language $\mathcal{L} \supseteq \mathcal{L}_A^2$ and let Φ be a set of formulas in the language $\mathcal{L}$. A number function f is Φ-definable in a theory $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, y, \vec{X})$ in Φ such that $\mathcal{T}$ proves

$$\forall \vec{x} \forall \vec{X} \exists! y \varphi(\vec{x}, y, \vec{X})$$

and it holds that

$$y = f(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, y, \vec{X}). \tag{B.1}$$

A string function F is Φ-definable in a theory $\mathcal{T}$ iff there is a formula $\varphi(\vec{x}, \vec{X}, Y)$ in Φ such that $\mathcal{T}$ proves

$$\forall \vec{x} \forall \vec{X} \exists! Y \varphi(\vec{x}, \vec{X}, Y)$$

and it holds that

$$Y = F(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}, Y). \tag{B.2}$$

Finally, a relation $R(\vec{x}, \vec{X})$ is Φ-definable iff there is a formula $\varphi(\vec{x}, \vec{X})$ in Φ such that it holds that

$$R(\vec{x}, \vec{X}) \leftrightarrow \varphi(\vec{x}, \vec{X}). \tag{B.3}$$

Moreover we wish to talk about sequences coded by strings or numbers. For a string X we let $X[i]$ be the ith bit of X. Assuming a tupling function $\langle \cdot, \ldots, \cdot \rangle$ we can also talk of k-ary relations in the obvious way by referring to $X[\langle i_0, \ldots, i_k \rangle]$. For the sake of simplicity we also refer to $X[\langle i_0, \ldots, i_k \rangle]$ by $X[i_0, \ldots, i_k]$.

Using k-ary relations we can also encode sequences of bounded numbers $x_0, \ldots, x_m$ by $x_i = X[\langle i, 0 \rangle] X[\langle i, 1 \rangle] \ldots X[\langle i, k \rangle]$ in binary. Matrices and so on can obviously be formalized in the same way.

Given a string $X[\langle x_1, \ldots, x_k \rangle]$ representing a k-ary relation, we denote the $k-\ell$-ary substring with parameters $a_{i_1}, \ldots, a_{i_\ell}$ by $X[\langle \cdot, \ldots, \cdot, a_{i_1}, \cdot, \ldots, a_{i_\ell}, \cdot, \ldots, \cdot \rangle]$. For example we refer to the element a_{ij} of a given matrix $A[\langle x_1, x_2, x_3 \rangle]$ as $A[\langle i, j, \cdot \rangle]$, a string representing a_{ij} in binary. Observe that this substring can be Σ_0^B defined in $\mathbf{V}^0$.

Given a number x we denote by $\langle x \rangle_j$ the jth number in the sequence encoded by x. To do this we assume a fixed Σ_0^B definable encoding of numbers that is 1-1.

The sequence itself will be addressed as $\langle x\rangle$. We can also talk about matrices, etc. in the way presented above.

We want to identify strings of short length with sequences of numbers. Thus, given a string X of length $O(n)$ we can Σ_0^B-define (in $\mathbf{V}^0$) a number $x \leq 2^{O(n)}$ that codes a sequence $\langle x\rangle$, such that $X[i] = \langle x\rangle_i$ for all $i < |X|$ and vice versa. We will use $\langle x\rangle \approx X$ and $\langle x\rangle_i \approx X[i]$ to denote the above identification. Observe that n has to be very small in order to be able to do the above in $\mathbf{V}^0$.

Computations in models of $\mathbf{V}^0$

Given a polynomially bounded Turing machine A in a binary encoding, we can Σ_1^B define a predicate $\mathsf{ACC}_A(X)$, that states that X is accepted by A. This can readily be observed, since, provided some machine A, there is a constant number of states $\sigma_1, \ldots, \sigma_k$ and the whole computation can be written into a matrix W of polynomial size. That W is indeed a correct computation can then be easily checked, because the computations are only local.

More precisely let $A = \langle \sigma_1, \ldots, \sigma_k; \delta\rangle$ be given, where the σ_i are different states, with σ_1 being the initial state and σ_k being the accepting state and δ is the transition function with domain $\{\sigma_1, \ldots, \sigma_k\} \times \{0, 1\}$ and range $\{\sigma_1, \ldots, \sigma_k\} \times \{0, 1\} \times \{\leftarrow, \downarrow, \rightarrow\}$, which describes what the machine does. I.e. if $\delta(a, b) = (c, d, e)$, then if the machine is in state a and reads b, it replaces b by d, goes into state c and moves one position on the tape in the direction e. For our formalization we will assume a function $\delta : \mathbb{N}^2 \to \mathbb{N}$ and interpret it in the following way, $\delta(\sigma_a, b) = (\sigma_{\langle\delta(a,b)\rangle_1}, \langle\delta(a, b)\rangle_2, \langle\delta(a, b)\rangle_3)$, where we identify $\downarrow = 0, \leftarrow = 1, \rightarrow = 2$.

Let the polynomial p bound the running time of A, then we can formalize $\mathsf{ACC}_A(X)$ as follows

$$
\begin{aligned}
\exists W \leq (p(|X|)^2 \cdot \log(k)^2)\forall i, i' \leq p(|X|)\forall 0 < \alpha < k(\\
&i < |X| \to (\langle W[\langle 0, i, \cdot\rangle]\rangle_1 = X[i] \wedge i > 0 \to \\
&\qquad \langle W[\langle 0, i, \cdot\rangle]\rangle_2 = 0 \wedge \langle W[\langle 0, 0, \cdot\rangle]\rangle_2 = 1)\wedge \\
&i \geq |X| \to (\langle W[\langle 0, i, \cdot\rangle]\rangle_1 = 0 \wedge \langle W[\langle 0, i, \cdot\rangle]\rangle_2 = 0)\wedge \\
&\langle\langle W[\langle j, i, \cdot\rangle]\rangle_2 = 0 \to (\langle W[\langle j + 1, i, \cdot\rangle]\rangle_1 = \langle W[\langle j, i, \cdot\rangle]\rangle_1)\wedge \\
&\langle W[\langle j, i, \cdot\rangle]\rangle_2 = \alpha \to (\langle W[\langle j + 1, i, \cdot\rangle]\rangle_1 = \langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_2\wedge \\
&\quad (\langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_3 = 0 \to \\
&\qquad \langle W[\langle j + 1, i, \cdot\rangle]\rangle_2 = \langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_1)\wedge \\
&\quad (\langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_3 = 1 \to \\
&\qquad \langle W[\langle j + 1, i \dot{-} 1, \cdot\rangle]\rangle_2 = \langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_1)\wedge \\
&\quad (\langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_3 = 2 \to \\
&\qquad \langle W[\langle j + 1, i + 1, \cdot\rangle]\rangle_2 = \langle\delta(\alpha, \langle W[\langle j, i, \cdot\rangle]\rangle_1)\rangle_1))\wedge \\
&i \neq i' \to (\langle W[\langle j, i, \cdot\rangle]\rangle_2 > 0 \to \langle W[\langle j, i', \cdot\rangle]\rangle_2 = 0)).
\end{aligned}
\tag{B.4}
$$

Thus, in plain English, $\mathsf{ACC}_A(X)$ says that there exists a matrix W of pairs of numbers that witnesses an accepting computation of A. Here, $\langle W[\langle i, j, \cdot\rangle]\rangle$ is supposed to code the jth cell on the Turing machine's tape after i steps of computations on input X. As noted above, $\langle W[\langle i, j, \cdot\rangle]\rangle_1$ is a binary number,

which is the value of the cell, $\langle W[\langle i, j, \cdot\rangle]\rangle_2$ is a number coding the state, the machine is in iff the pointer is on that cell.

The second and third line of the definition say that the tape in the initial step contains X padded with zeroes in the end to get the proper length ($p(|X|)$). The fourth line says that if the read/write head is not on cell i, then nothing happens to the content of cell i. The fourth line says that the content of the cell, where the read/write head is in step j, is changed according to δ. The next three lines tell us where the read/write head moves and the last line says that there is at most one position on the tape where the read/write head may be at any step.

We also define a Σ_1^B-predicate $\mathsf{REACH}_A(X, Y)$ that says that A reaches configuration Y from configuration X in at most $p(|X|)$ steps. This is essentially the same predicate as ACC, with the constraints on the initial and accepting state lifted and instead a constraint added that the first line of computation is X and the last is Y. We omit the details as it does not severely differ from the above definition of ACC.

B.2.3 Extensions of $\mathbf{V}^0$

The Theory $\mathbf{V}^0$ serves as our base theory to describe complexity classes by arithmetical means.

The problem, whether a given monotone formula φ of size ℓ and depth $\lceil\log(\ell)\rceil$ is satisfiable under a given assignment I is AC^0-complete for NC^1. Therefore Cook and Nguyen ([29]) define the class $\mathbf{VNC}^1$ as $\mathbf{V}^0$ augmented by the axiom $MFV \equiv \exists Y \leq 2a + 1.\delta_{MFV}(a, G, I, Y)$, where

$$\begin{aligned}\delta_{MFV}(a, G, I, Y) \equiv \forall x < a((Y(x + a) \leftrightarrow I(x)) \wedge Y(0) \wedge \\ 0 < x \rightarrow (Y(x) \leftrightarrow ((G(x) \wedge Y(2x) \wedge Y(2x + 1)) \vee \\ (\neg G(x) \wedge (Y(2x) \vee Y(2x + 1)))))).\end{aligned}$$

So, MFV states that there is an evaluation Y of the monotone formula represented by G under the assignment given by I of length at most $2a + 1$. More specifically, G is a tree-encoding of the formula, where $G(x)$ is true, if node x is $\wedge$ and false, if x is $\vee$. The evaluation Y takes the value of the variables given by I and then evaluates the formula in a bottom-up fashion using a standard tree encoding. Thus, the value of the formula can be read at $Y(1)$.

It is interesting to observe that MFV does not hold in $\mathbf{V}^0$, as, by the Witnessing Theorem for $\mathbf{V}^0$, a Σ_0^B definition of the satisfaction relation would be sufficient to prove that $\mathsf{NC}^1 \subseteq \mathsf{AC}^0$, at least for monotone functions, which is known to be false.

B.2.4 Relation between Arithmetic Theories and Proof Systems

In this section we will remind the reader of a connection between the Theory $\mathbf{V}^0$ and some of its extensions and certain propositional proof systems (see also [29][57]).

Definition B.5. The following predicates will be subsequently used. They are definable with respect to $\mathbf{V}^0$ (see [57]).

- $\mathsf{Fla}(X)$ is a Σ_0^B formula that says that the the string X codes a formula.
- $Z \models X$ is the Δ_1^B definable property that the truth assignment Z satisfies the formula X.
- $\mathsf{Taut}(X)$ is the Π_1^B formula $\mathsf{Fla}(X) \wedge \forall Z \leq t(|X|)Z \models X$, where t is a number term.
- $\mathsf{Prf}_{F_d}(\Pi, A)$ is a Σ_0^B definable predicate meaning Π is a depth d Frege proof for A.
- $\mathsf{Prf}_F(\Pi, A)$ is a Σ_0^B definable predicate meaning Π is a Frege proof for A.

Observe that in $\mathbf{V}^0$ we cannot prove that every formula has an evaluation. The following holds

Fact 5 (see [29])**.** The Theory $\mathbf{V}^0$ proves that AC^0-Frege is sound, i.e. for every d

$$\forall A \forall \Pi \mathsf{Prf}_{F_d}(\Pi, A) \rightarrow \mathsf{Taut}(A).$$

Fact 6 (see [29])**.** The Theory $\mathbf{VNC}^1$ proves that Frege is sound, i.e.

$$\forall A \forall \Pi \mathsf{Prf}_F(\Pi, A) \rightarrow \mathsf{Taut}(A).$$

On the other hand, provability of the universal closure of Σ_0^B formulas in $\mathbf{V}^0$ and $\mathbf{VNC}^1$ implies the existence of polynomial size proofs of their propositional translations in AC^0-Frege and Frege, respectively.

The propositional translation $[\![\varphi(\bar{x}, \bar{X})]\!]_{\bar{m},\bar{n}}$ of a Σ_0^B formula $\varphi(\bar{x}, \bar{X})$ is a family of propositional formulas built up inductively (on the logical depth) as follows. If φ is atomic, then we evaluate φ in $\mathbb{N}_2$, if it contains second sort variables, we have to introduce propositional variables. If φ is a boolean combination of formulas ψ_i of lower depth, the translation is simply the same boolean combination of the translations of the ψ_i. If φ is $\exists\psi$ or $\forall\psi$ we translate it to the disjunction or conjunction of the translations, respectively. For a proper definition see [29].

Fact 7. There exists a polynomial p such that for all Σ_0^B formulas $\varphi(\bar{x}, \bar{X})$ the following holds

- If $\mathbf{V}^0 \vdash \forall \bar{X} \forall \bar{x} \varphi(\bar{x}, \bar{X})$, then there exist bounded depth Frege proofs of all $[\![\varphi]\!]_{\bar{m},\bar{n}}$ of length at most $p(\max(\bar{m}, \bar{n}))$, for any $\bar{m}, \bar{n}$.
- If $\mathbf{VNC}^1 \vdash \forall \bar{X} \forall \bar{x} \varphi(\bar{x}, \bar{X})$, then there exist Frege proofs of all $[\![\varphi]\!]_{\bar{m},\bar{n}}$ of length at most $p(\max(\bar{m}, \bar{n}))$, for any $\bar{m}, \bar{n}$.

Facts 5 and 6 are examples of general principles, the so called *Reflection Principles*, which are defined as follows.

Definition B.6 (Reflection Principle)**.** Let P be a pps. Then the *Reflection Principle* for P, Ref_P, is the $\forall\Delta_1^B$-formula (w.r.t. $\mathbf{V}^0$)

$$\forall \Pi \forall X \forall Z((\mathsf{Fla}(X) \wedge \mathsf{Prf}_P(\Pi, X)) \rightarrow (Z \vDash X)),$$

where Prf_P is a Δ_1^B-predicate formalizing P-proofs.

Reflection Principles condense the strength of propositional proof systems. In what follows we will summarize some such results for the proof systems and theories used here. A detailed exposition can be found in [29], chapter X, or in [57], chapter 9.3.

Theorem B.7. *If* $\mathbf{V}^0 \vdash \mathsf{Ref}_F$ *then bounded depth Frege p-simulates Frege.*

We will only give a brief sketch of the proof here and leave out the technical details.

Sketch. Let φ be a formula and π_φ a Frege proof of φ. Since $\mathbf{V}^0$ proves Ref_F, by Facts 5 and 7 we have polynomial size proofs of its translations $[\![\mathsf{Ref}_F]\!]$ in bounded depth Frege. Bounded depth Frege itself, however, is strong enough to verify that a proper encoding of the computation of the Turing machine verifying the Frege proof π_φ is correct. Thus it can verify that π_φ is a Frege-proof and, using the translation of the Reflection Principle and the Cut rule, conclude $[\![\mathsf{Taut}(\varphi)]\!]$. From this φ follows, cf. [57] Lemma 9.3.7.

□

Given a term t and a variable x, we can also introduce the t-bounded version of the Reflection Principle for some given pps P, $\mathsf{Ref}_P(t(x))$ that claims soundness only for t-bounded proofs.

Definition B.8 (Bounded Reflection)**.** Let t be a $\mathcal{L}^2_{Ar}$-term, x a first-sort variable and P a pps. Then the *Bounded Reflection Principle* $\mathsf{Ref}_P(t(x))$ is the formula

$$\forall \Pi \leq t(x) \forall X \leq t(x) \forall Z \leq t(x)((\mathsf{Fla}(X) \wedge \mathsf{Prf}_P(\Pi, X)) \rightarrow (Z \vDash X)).$$

We can now generalize Theorem B.7 in the following way.

Theorem B.9. *Let* t *be a* $\mathcal{L}^2_A$*-term and* x *a number variable. If* $t(x) < x$ *for* x *large enough and if* $\mathbf{V}^0 \vdash \forall x \mathsf{Ref}_F(t(x))$ *then for every propositional formula* φ *with a Frege proof of length* $t(x)$ *there is a bounded depth Frege proof of* φ *of length* $x^{O(1)}$*.*

Proof. The proof is the same as that of Theorem B.7. Using the Bounded Reflection Principle we can encode Frege proofs of length $t(x)$ as bounded depth Frege proofs of length $x^{O(1)}$. □

As a corollary we get

Corollary B.10. If $\mathbf{V}^0 \vdash \mathsf{Ref}_F(|x|^k)$ for all $k \in \mathbb{N}$, then bounded depth Frege sub exponentially simulates Frege: For all $D > 1, \delta > 0$ exists $d \geq D$, such that the existence of a Frege proof of length m of a depth D formula implies the existence of a depth d Frege proof of length at most 2^{m^δ}.

B.3 Polylogarithmic Cuts of Models of $\mathbf{V}^0$ are Models of $\mathbf{VNC}^1$.

We will first introduce the notion of a cut $\mathcal{I}$ of a given two-sorted arithmetic model $\mathcal{M}$. This model theoretic approach provides a very good insight on what actually happens semantically with the small elements of arithmetical models.

Definition B.11 (Cut)**.** Let T be a two-sorted arithmetic theory and

$$N = \{N_1, N_2, +^N, \cdot^N, \leq^N, 0^N, 1^N, |\cdot|^N, =_1^N, =_2^N, \in^N\}$$

a model of T. A *cut*

$$M = \{M_1, M_2, +^M, \cdot^M, \leq^M, 0^M, 1^M, |\cdot|^M, =_1^M, =_2^M, \in^M\}$$

of N is any substructure such that

- $M_1 \subseteq N_1$, $M_2 \subseteq N_2$,
- $0^M = 0^N$, $1^M = 1^N$,
- M_1 is closed under $+^N, \cdot^N$ and downwards with respect to $\leq^N$,
- $M_2 = \{X \in N_2 \mid X \subseteq M_1\}$, and
- $\circ^M$ is the restriction of $\circ^N$ to M_1 and M_2 for all relation and function symbols $\circ \in \mathcal{L}_A^2$.

We call this cut the *Polylogarithmic Cut* iff

$$x \in M_1 \Leftrightarrow \exists a \in N_1, k \in \mathbb{N}\ x \leq |a|^k .$$

To examine the strength of the theory of such cuts of models of $\mathbf{V}^0$, we will show that a formal connection between efficient computability and Σ_0^B-definability holds. This stands in contrast to general bounded subsets, where the connection is presumably only with respect to Σ_1^B-definability via the predicate ACC (see (B.4) on page 67). The intended theorem is a formalization of Nepomnjascij's Theorem [68] (see also [57] pg.20). We will sketch the original proof before starting the formalization.

Theorem B.12 (Nepomnjascij [68])**.** *Let $c \in \mathbb{N}$ and $0 < \epsilon < 1$ be constants. Then if the language $L \in$ **TimeSpace**(n^c, n^ϵ), the relation $x \in L$ is definable by a Σ_0^B-formula over $\mathbb{N}$.*

Proof. We will prove the theorem by induction on k for $L \in$ **TimeSpace**$(n^{k\cdot(1-\epsilon)}, n^\epsilon)$.

Let $k = 1$ and $L \in$ **TimeSpace**$(n^{k\cdot(1-\epsilon)}, n^\epsilon)$. For any $x \leq 2^n$ the whole computation can be coded by a number y of size $2^{O(n)}$.

For $k > 1$ we write a sequence $y_0, y_1, \ldots, y_{n^{1-\epsilon}}$ of intermediate results coding the computation, where y_0 codes the starting configuration on input x, such that we can verify that y_{i+1} is computable from y_i in **TimeSpace**$(n^{(k-1)\cdot(1-\epsilon)}, n^\epsilon)$. Therefore by assumption there exists a Σ_0^B-formula reach^{k-1} such that $\mathsf{reach}^{k-1}(y_i, y_{i+1})$ holds iff y_{i+1} is computed from y_i. Additionally, the whole sequence has length $O(n)$ and so we can write the sequence of intermediate results y_i as a number y of length $O(n)$. Now, the Σ_0^B-definition of $x \in L$ is simply

$$\begin{aligned}\exists y \leq 2^{O(n)} \forall i \leq n^{1-\epsilon} &\mathsf{reach}^{k-1}(\langle y\rangle_i, \langle y\rangle_{i+1}) \\ &\wedge \langle y\rangle_0 \text{ encodes the starting configuration of } A \text{ on input } x \\ &\wedge \langle y\rangle_{n^{1-\epsilon}} \text{ is in an accepting state.}\end{aligned}$$

□

We will now formalize this result in $\mathbf{V}^0$ as follows

Theorem B.13. *Let $N \vDash \mathbf{V}^0$. Let $m = |a|$ for some $a \in N_1$ and let $c, k \in \mathbb{N}$ and $\epsilon < 1$. If $L \in \mathsf{TimeSpace}(m^c, m^\epsilon)$ is computed by Turing machine A, then there exists a Σ^B_0 definition in N of the Σ^B_1-predicate ACC_A on the interval $[0, m^k]$. I.e. any $Y \in L$, bounded by m^k is Σ^B_0-definable in N and therefore exists in the polylogarithmic cut of N.*

The following version of the proof stems from a discussion with Stephen Cook and Neil Thapen during the SAS programme in Cambridge. It is more explicit than the original one and clarifies the argument.

Proof. We will inductively on d define a Σ^B_0 relation $\mathsf{reach}^d_A(I, p_1, p_2, cell, \mathsf{comp})$ that states that the p_2th cell of the work tape of A, starting on configuration I and computing for $p_1 \cdot m^{d\frac{1-\epsilon}{k}}$ steps via the computation comp is $cell$. As d depends only on A and k we will be doing this induction outside of the theory to construct d many formulas. We will then prove the above mentioned properties of reach_A by Σ^B_0 induction on p_1.

Keep in mind that a cell is given as a pair $\langle bit, state\rangle$, where bit is the actual value of the cell and $state$ is a constant binary number > 0 coding the state the Turing machine is in iff the pointer is on that cell and 0 otherwise. As before the transition function is denoted by δ. We let $|I| = m^k$.

To this end let $\mathsf{reach}^0_A(I, p_1, p_2, cell, \mathsf{comp}) \equiv$

$$
\begin{aligned}
&(\forall j' < \lceil |I|^\epsilon \rceil, j'' < |A|\ \langle \mathsf{comp}\rangle_{\langle 1,j',j''\rangle} \leftrightarrow \langle I\rangle_{\langle j',j''\rangle})\wedge\\
&(\forall j < \lceil |I|^{\frac{1-\epsilon}{k}} \rceil, j' < \lceil |I|^\epsilon \rceil, \alpha < |A|\\
&(\langle \mathsf{comp}\rangle_{\langle j,j',\cdot\rangle} = 0 \rightarrow (\langle \mathsf{comp}\rangle_{\langle j,j',0\rangle} \leftrightarrow \langle \mathsf{comp}\rangle_{\langle j+1,j',0\rangle}))\wedge\\
&(\langle \mathsf{comp}\rangle_{\langle j,j',\cdot\rangle} = \alpha \rightarrow (\\
&(\langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_3 = 0 \rightarrow (\langle \mathsf{comp}\rangle_{\langle j+1,j',\cdot\rangle} = \langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_1\wedge\\
&\langle \mathsf{comp}\rangle_{\langle j+1,j',0\rangle} = \langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_2))\wedge\\
&(\langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_3 = 1 \rightarrow (\langle \mathsf{comp}\rangle_{\langle j+1,j'\dot{-}1,\cdot\rangle} = \langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_1\wedge\\
&\langle \mathsf{comp}\rangle_{\langle j+1,j',0\rangle} = \langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_2))\wedge\\
&(\langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_3 = 2 \rightarrow (\langle \mathsf{comp}\rangle_{\langle j+1,j'+1,\cdot\rangle} = \langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_1\wedge\\
&\langle \mathsf{comp}\rangle_{\langle j+1,j',0\rangle} = \langle \delta(\alpha, \langle \mathsf{comp}\rangle_{\langle j,j',0\rangle})\rangle_2))))\wedge\\
&\forall \ell, \ell' < \lceil |I|^\epsilon \rceil(\langle \mathsf{comp}\rangle_{\langle j,\ell,\cdot\rangle} > 0 \rightarrow \langle \mathsf{comp}\rangle_{\langle j,\ell',\cdot\rangle} = 0))\wedge\\
&(\langle cell\rangle_1 \leftrightarrow \langle \mathsf{comp}\rangle_{\langle p_1,p_2,0\rangle}) \wedge (\langle cell\rangle_2 = \langle \mathsf{comp}\rangle_{\langle p_1,p_2,\cdot\rangle}).
\end{aligned}
$$

Observe that above we abused notation by writing $\langle \mathsf{comp}\rangle_{\langle j,j',\cdot\rangle} = \alpha$, when we actually meant that the sequence of bits $\langle \mathsf{comp}\rangle_{\langle j,j',1\rangle}, \ldots, \langle \mathsf{comp}\rangle_{\langle j,j',|A|\rangle}$ is the binary representation of α. We have left out $\langle \mathsf{comp}\rangle_{\langle j,j',0\rangle}$ as this is the actual value of the cell.

It is straightforward to prove by induction on the number of lines in comp that comp is uniquely defined by reach^0_A. We let

$$\mathsf{Reach}^0_A(I, p_1, p_2, cell) =_{def} \exists \mathsf{comp} < q(|I|)\ \mathsf{reach}^0_A(I, p_1, p_2, cell, \mathsf{comp}),$$

where q is some polynomial depending on the encoding.

We will now proceed by inductively defining reach^d_A and Reach^d_A. Assume that $\mathsf{reach}^{d\dot{-}1}_A$ has already been defined. We then let
$\mathsf{reach}^d_A(I, p_1, p_2, cell, \mathsf{comp}) \equiv$

$$
\begin{aligned}
&(\forall j' < \lceil |I|^{\epsilon} \rceil, j'' < |A|\, \langle \mathsf{comp} \rangle_{\langle 1, j', j'' \rangle} \leftrightarrow \langle I \rangle_{\langle j', j'' \rangle}) \wedge \\
&(\forall j < \lceil |I|^{\frac{1-\epsilon}{k}} \rceil \exists \mathsf{comp}' < q(|I|) \forall j' < \lceil |I|^{\epsilon} \rceil \exists cell' < |A|\, \forall j'' < |A| \\
&(\langle \mathsf{comp} \rangle_{\langle j+1, j', j'' \rangle} \leftrightarrow \langle cell' \rangle_{j''}) \wedge \\
&\mathsf{reach}^{d\dot{-}1}_A(\langle \mathsf{comp} \rangle_{\langle j, \cdot, \cdot \rangle}, m^{\frac{1-\epsilon}{k}}, j', cell', \mathsf{comp}')) \wedge \\
&(\langle cell \rangle_1 \leftrightarrow \langle \mathsf{comp} \rangle_{\langle p_1, p_2, 0 \rangle}) \wedge (\langle cell \rangle_2 = \langle \mathsf{comp} \rangle_{\langle p_1, p_2, \cdot \rangle}).
\end{aligned}
$$

Again, we can prove uniqueness of the computation by induction on the number of its lines and let

$$\mathsf{Reach}^d_A(i, p_1, p_2, cell) =_{def} \exists \mathsf{comp} < q(|I|)\ \mathsf{reach}^d_A(i, p_1, p_2, cell, \mathsf{comp}).$$

We now can give a Σ^B_0 definition of the predicate $W[\langle i, j, \cdot \rangle]$ coding the computation as in ACC_A on input X of length m^k. We let

$$
\begin{aligned}
W[\langle i, j, \cdot \rangle] = cell \equiv \exists r_0, \ldots, r_d < |X|^{\frac{1-\epsilon}{k}}, con_1, \ldots, con_d < p(|X|^{\epsilon}) \\
\forall z_1, \ldots, z_d < |X|^{\epsilon}\, \exists cell_1, \ldots, cell_d < |A| \\
(i = \sum_{\ell=0}^{d} r_\ell \cdot |X|^{\ell \frac{1-\epsilon}{k}} \\
\wedge\, \mathsf{Reach}^d_A(\tilde{X}, r_d, z_d, cell_d) \wedge \langle con_d \rangle_{z_d} = cell_d \\
\wedge\, \mathsf{Reach}^{d-1}_A(con_d, r_{d-1}, z_{d-1}, cell_d) \wedge \langle con_{d-1} \rangle_{z_{d-1}} = cell_{d-1} \\
\vdots \\
\wedge\, \mathsf{Reach}^0_A(con_1, r_0, j, cell)),
\end{aligned}
$$

where p is a polynomial depending on the encoding and $\tilde{X}$ is the starting configuration of A on input X.

Informally the above formula says that we compute the configurations of A by using the predicates Reach^d_A through Reach^0_A. That is, after the application of Reach^d_A (i.e. after making the biggest steps) we have reached configuration con_d, which we plug into Reach^{d-1}_A to get configuration con_{d-1} and so on. It remains to show that this definition of $W[\langle i, j, \cdot \rangle]$ coincides with the real one, i.e. that $W[\langle i+1, \cdot, \cdot \rangle]$ follows from an application of the transition function of A from $W[\langle i, \cdot, \cdot \rangle]$.

We will prove this inductively, depending on i. Again let r_ℓ be such that $i = \sum_\ell r_\ell \cdot |X|^{\ell \frac{1-\epsilon}{k}}$. If $i < |X|^{\frac{1-\epsilon}{k}}$ the assumption follows straightforwardly from the definition of Reach^0_A. Now for bigger i. If the r_0, given as above, is bigger then 0 then again the assumption follows from the definition of reach^0_A. Now let $\ell > 0$ be the first index with $r_\ell > 0$. We the have to argue that $\mathsf{reach}^{\ell'-1}_A$ has the desired property. This, however, follows straightforward if we can verify this assertion for $\mathsf{reach}^{\ell'-1}_A$. Observe that d is a constant depending only on A and k, so we need to make this argument only a constant number of steps to reach reach^0_A, where we know that the assertion holds. This concludes the proof. □

We can now prove our main result.

Theorem B.14. *Let $N \models \mathbf{V}^0$ and $M \subseteq N$ be the polylogarithmic cut. Then $M \models \mathbf{VNC}^1$.*

Proof. We have to prove that for all strings $G_\varphi \in M_2$, representing a formula φ as a tree and assignments $I \in M_2$ to its variables (i.e. leafs in the tree representation) a string Y exists in M_2 that contains all values of φ's subformulas as in the definition of MFV in Section B.2.3 and satisfies the inductive conditions of MFV.

However, by Σ_0^B-comprehension and the formalized Nepomnjascij's Theorem it suffices to describe an algorithm that computes whether $i \in Y$ from G_φ and I in $\mathsf{TimeSpace}(|G_\varphi|^k, |G_\varphi|^\epsilon)$ for some $k \in \mathbb{N}$ and $\epsilon < 1$. To this end let G_φ and I be given.

The following is a recursive algorithm computing the value of $Y[i]$, given $G := G_\varphi, I$ and i.

```
NodeValue(G,I,i)
```

- `boolean left; boolean right;`
- `If i>2·|G|`
  - `Output (0); End;`
- `Else If i>|G|`
  - `Output (I[i-|G|]); End;`
- `Else If G[i]=1`
  - `left := NodeVal(G,I,2i);`
  - `right := NodeVal(G,I,2i+1);`
  - `Output (left AND right); End;`
- `Else If G[i]=0`
  - `left := NodeVal(G,I,2i);`
  - `right := NodeVal(G,I,2i+1);`
  - `Output (left OR right); End;`
- `Else`
  - `Output (0); End;`

Observe that the algorithm at any given point only stores a constant amount of data per level of the tree G and therefore uses only $O(\log(|G|))$ space. The number steps the algorithm makes is clearly polynomial in the size of G. Therefore by Theorem B.13, for every monotone formula φ, representable as a tree in M, we get a Σ_0^B formula eval_φ, such that $\mathsf{eval}_\varphi(i, I) \equiv Y[i]$. Applying the Comprehension Schema in $\mathbf{V}^0$, i.e. in N, this verifies the existence of a Y as in MFV for all formulas represented by trees in M.Therefore MFV holds in M and so $M \models \mathbf{VNC}^1$.

□

B.4 Implications for Proof Complexity

We now wish to apply the above results to propositional proof systems. More precisely we wish to show that theories of small cuts of a model of a given theory $\mathcal{T}$ correspond to stronger proof systems than $\mathcal{T}$ does. An elegant way of showing such a statement is via the *Reflection Principles* of the given proof systems, i.e. the statement that the proof system is correct, as explained in Section B.2.4. With their help we can conclude the following recent result of Filmus, Pitassi and Santhanam [39].

Theorem B.15 ([39])**.** *Every Frege system is sub exponentially simulated by* AC^0*-Frege systems.*

Proof. By Theorem B.9 we have to prove the polylogarithmically bounded Reflection Principle for Frege in $\mathbf{V}^0$. This, by Theorem B.14 however, corresponds to proving the Reflection Principle for Frege in $\mathbf{VNC}^1$, which holds by Fact 6. □

Another, related, application is in the separation of propositional proof systems. In [67] we proved the following.

Proposition B.16. For almost every random 3CNF A with n variables and $m = c \cdot n^{1,4}$ clauses, where c is a large constant, $\neg A$ has polynomially bounded TC^0-Frege proofs.

On the other hand it is well known (see for example [25]) that such formulas have no subexponential refutation in Resolution. Thus, this yields an average case separation between Resolution and TC^0-Frege. We can now extend this result to an average case separation between Resolution and AC^0-Frege as follows.

Theorem B.17. *For almost every random 3CNF A with n variables and $m = c \cdot n^{1,4}$ clauses, where c is a large constant, $\neg A$ has subexponentially bounded* AC^0*-Frege proofs.*

Proof. By Theorem B.14 the polylogarithmic Cut of any $\mathbf{V}^0$-model is a model of $\mathbf{VNC}^1$, therefore also of $\mathbf{VTC}^0$. This yields, as in our proof of Theorem B.15, that AC^0-Frege subexponentially simulates TC^0-Frege. The result now follows from Proposition B.16. □

B.5 Conclusion and Discussion

As we have seen cuts of models of weak arithmetics constitute an appropriate way for reasoning about super-polynomial simulations between proof systems. An advantage in comparison to syntactic arguments is the possible applicability of results in Model Theory and a more uniform treatment. This can readily be observed as with our argument, e.g. the work of Paris and Wilkie [74][75] immediately imply the simulation results from Bonet et al. [17].

This leads to interesting possibilities for further research, especially towards the weak automatizability of weak propositional proof systems such as Resolution. The underlying theory, which was $\mathbf{V}^0$ in our argument, must be significantly weakened, however. If we could take $T^2_1(\alpha)$ as our base theory, we

could reason about whether $Res(\log)$ has the feasible interpolation property in the same way as Krajíček and Pudlák [61], Bonet, Pitassi and Raz [18] or Bonet, Domingo, Gavaldà, Maciel, and Pitassi [17]. Now, if $Res(\log)$ does not have quasi-polynomial feasible interpolation we know by a result from Atserias and Bonet [6] that Resolution is not weakly automatizable, so we would be finished. Whether we can actually do it depends on the strength of the theory the polylogarithmic cut of $T_1^2(\alpha)$ models and if we can formalize some sort of iterated multiplication (such as in [51]) in that theory. Also, the security of Diffie-Hellman seems to be a more appropriate assumption than that of RSA, as the computational power needed to verify the correctness of Diffie-Hellman seems to be lower.

B.6 Acknowledgements

I want to thank Steve Cook, Jan Krajíček and Neil Thapen for helpful suggestions and discussion, Emil Jeřábek for his comments and for answering my questions and the participants of the MALOA Special Semester in Proof Complexity in Prague 2011 for enduring a sloppy and sometimes faulty exposition of this proof and still coming up with helpful comments. I also want to thank the anonymous referees for pointing out various mistakes and for giving interesting suggestions.

C. Necessary Background

Here we will summarize and briefly explain the logical and complexity theoretic concepts we need.

C.1 Logical Preliminaries

Historically, logic started to become more influential in mathematics with the advent of methods which are not "obviously true". At latest from the 19th century on mathematical arguments became more abstract and in many cases used objects which were not rigorously constructed. A famous early example is Dirichlet's definition of a function that is 1 on the rational and 0 on the irrational numbers. This was revolutionary at its time as its definition did not provide a clear means of computing the image. Aside from leading to various new results, use of such methods also bore a great risk of inconsistency. This let to various criticisms of such arguments as to whether these results should be acceptable at all. David Hilbert, one of the strongest proponents of the use of such non-finitary techniques, tried to counter such criticism in his famous Conservation Program: To eliminate the above problem, can we give an explicit collection of feasible true statements and a system of rules, which allows us to manipulate sentences in a way that does not alter the truth value of true sentences, such that all statements about finitary objects proved in an abstract way can be reproved using that system? Hilbert himself believed that a fragment of elementary number theory exists that would be appropriate as a base for an axiomatization. This fragment should be consisting of "gewisse außerlogische diskrete Objekte, die anschaulich als unmittelbares Erlebnis vor allem Denken da sind. Soll das logische Schließen sicher sein, so müssen sich diese Objekte vollkommen in allen Teilen überblicken lassen und ihre Aufweisung, ihre Unterscheidung, ihr Aufeinanderfolgen ist mit den Objekten zugleich unmittelbar anschaulich für uns da als etwas, das sich nicht noch auf etwas anderes reduzieren läßt."[1] [52]. That this view is not correct was shown shortly after by Kurt Gödel as we will see in a moment.

Hilbert's program became the driving force of introducing a rigorous logical framework into mathematics. Although its original goal proved to be impossible to achieve, it became extremely successful in analyzing and structuring mathematics.

For now, we will call such a system of deduction a calculus and such an explicit collection of statements a theory. To treat the above question we need to talk about objects that live in a mathematical world, e.g. the elements of the set of natural numbers. We will, however, start with easier objects to cope with. That is, we will start with statements as primitives and ways to reason about them. An excellent and more extensive survey can be found in [22] or with a different focus in [35].

[1] "extra-logical discrete objects, which exist intuitively as immediate experience before all thought. If logical inference is to be certain, then these objects must be capable of being completely surveyed in all their parts, and their presentation, their difference, their succession (like the objects themselves) must exist for us immediately, intuitively, as something which cannot be reduced to something else."

C.1.1 Propositional Logic

We will briefly sketch *propositional* or *sentential logic*. In such a logic our primitive objects of concern are propositions. We will make certain assumptions about how we want to evaluate these propositions and their interconnectivity. The first principle we want to adhere to is the principle of *bivalence*, that is, every statement is either *true* or *false*. The other principle is the principle of *extensionality*. That is, the truth value of any statement is determined by its propositional variables. We will soon formalize these principles in various *calculi*. But to do so, we first have to formalize our language.

We choose a very simply language to reason about. That is, we allow the propositions only to be connected by the connectives *and*, *or* and *not* and have a symbol for *true* and *false*. Formally, we will consider the symbols $\wedge$, $\vee$ and $\neg$, standing for conjunction, disjunction and negation, as logical connectives and let $\top$ and $\bot$ be constant symbols representing true and false, respectively. We denote propositional variables by lower case letters $p, q, r, \ldots$, possibly indexed.

A *propositional formula* is precisely any object that can be constructed in finitely many steps using the following rules.

- Propositional variables are propositional formulas.
- $\top$ and $\bot$ are propositional formulas.
- If φ, ψ are propositional formulas, then so are $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ and $(\neg\varphi)$.

We will omit unnecessary brackets and will agree on that $\neg$ binds stronger than $\vee$ and $\wedge$.

To incorporate the principle of extensionality we evaluate a formula by assigning a value to each of its variables and then successively evaluate the whole statement by applying boolean functions corresponding to the connectives to the values of the subformulas. We let

$$\begin{aligned} \mathsf{AND} : \{\bot,\top\} \times \{\bot,\top\} &\longrightarrow \{\bot,\top\} \\ (\bot,\bot) &\mapsto \bot \\ (\top,\bot) &\mapsto \bot \\ (\bot,\top) &\mapsto \bot \\ (\top,\top) &\mapsto \top \end{aligned}$$

$$\begin{aligned} \mathsf{OR} : \{\bot,\top\} \times \{\bot,\top\} &\longrightarrow \{\bot,\top\} \\ (\bot,\bot) &\mapsto \bot \\ (\top,\bot) &\mapsto \top \\ (\bot,\top) &\mapsto \top \\ (\top,\top) &\mapsto \top \end{aligned}$$

$$\begin{aligned} \mathsf{NOT} : \{\bot,\top\} &\longrightarrow \{\bot,\top\} \\ \bot &\mapsto \top \\ \top &\mapsto \bot \end{aligned}$$

If the outcome of the above process for some assignment w to α is $\top$ we say that w *satisfies* α and denote that symbolically as $w \vDash \alpha$. Two formulas α, β are *equivalent* (in symbols $\alpha \equiv \beta$) iff they have the same truth values under all assignments, i.e. iff for all assignments w to the variables of α and β it holds that $w \vDash \alpha \Leftrightarrow w \vDash \beta$. We will write $\vDash \alpha$ if a formula is satisfied by every assignment and call such a formula a *tautology*. Let Φ be a set of formulas and φ be a formula. We say Φ *entails* φ, iff every assignment that satisfies all formulas in Φ also satisfies φ.

We can now prove the following rules via application of the above boolean functions inductively on the nesting of the connectives in each formula.

Associativity.	$\alpha \wedge (\beta \wedge \gamma) \equiv \alpha \wedge \beta \wedge \gamma$ and
	$\alpha \vee (\beta \vee \gamma) \equiv \alpha \vee \beta \vee \gamma$
Commutativity.	$\alpha \wedge \beta \equiv \beta \wedge \alpha$ and
	$\alpha \vee \beta \equiv \beta \vee \alpha$
Idempotency.	$\alpha \wedge \alpha \equiv \alpha$ and
	$\alpha \vee \alpha \equiv \alpha$
Absorption.	$\alpha \wedge (\alpha \vee \beta) \equiv \alpha$ and
	$\alpha \vee (\alpha \wedge \beta) \equiv \alpha$
Distributivity.	$\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$ and
	$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
De Morgan Rules.	$\neg(\alpha \wedge \gamma) \equiv \neg\alpha \vee \neg\gamma$ and
	$\neg(\alpha \vee \gamma) \equiv \neg\alpha \wedge \neg\gamma$

From now on we will use abbreviations, such as $p \rightarrow q$ for $\neg p \vee q$, freely in our formulas. As is standard, we will identify 0 with $\bot$ and 1 with $\top$.

We proceed by turning the above semantic discussion of propositional logic into a syntactical one. I.e. we will give calculi that allow to derive tautologies without having to evaluate them. We call such a calculus C *sound*, iff every formula, derivable in C, is a tautology. We call it *implicationally complete*, iff, whenever Φ entails φ, there is a derivation of φ in C, from formulas in Φ. We will now give examples of such calculi and specify what we mean with derivability in such systems.

Hilbert- or Frege Calculi

A Hilbert- or Frege calculus consists of *rules* and *axioms*. We will stick to the name Frege system for such calculi now. This name was coined by Cook and Reckhow in [30].

A Frege rule is written as

$$\frac{A_1 \dots A_k}{A_0} \text{ (Frege Rule)}$$

and consists of proposition formulas $A_0 \dots A_k$. $A_1 \dots A_k$ are the *premisses* of the rule, A_0 the *conclusion*. An axiom is simply a rule with $k = 0$. A formula φ_0 is *inferred* from formulas $\varphi_1 \dots \varphi_k$ iff each φ_i is a substitution instance of A_i, obtained by the same substitution.

A *derivation* of a formula φ in such a system is a sequence $S_0, \dots, S_k$, where φ is a substitution instance of S_k, such that each S_i is an axiom or inferred from previous elements. A derivation from Φ of a formula φ is a derivation where we additionally allow the appearance of formulas from Φ. We call k the *length* of the derivation and each S_i a *proof line*. We use the symbol $\vdash^F \varphi$ to denote that φ is derivable in a Frege system and $\Phi \vdash^F \varphi$ to denote derivability from Φ.

Such a system F is a Frege system iff

- F is sound and
- F is implicationally closed.

It is noteworthy that it suffices to have *Modus Ponens* as the only inference rule.

The following is Frege's Propositional Calculus PC.

It consists of the rule of Modus Ponens

$$\frac{B \qquad B \to A}{A} \text{ (MP)}$$

and the axioms

Frege 1.	$A \to \neg\neg A$
Frege 2.	$\neg\neg A \to A$
Frege 3.	$(A \to B) \to (\neg B \to \neg A)$
Frege 4.	$A \to (B \to A)$
Frege 5.	$(A \to (B \to C)) \to (B \to (A \to C))$
Frege 6.	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

The Frege calculus satisfies a completeness theorem.

Theorem C.1. *The Frege calculus is complete, i.e. for all propositional formulas φ we have that*

$$\vDash \varphi \;\Leftrightarrow \vdash^F \varphi.$$

Gentzen's Sequent Calculus

A from a proof theoretic perspective very important calculus is Gentzen's propositional calculus PK developed in [43, 44]. It is a very elegant calculus allowing for deep proof-theoretic analysis and we refer to [57] for a more thorough introduction. PK is defined from sequents $\Gamma \longrightarrow \Delta$, where Γ and Δ are sequences of formulas. Semantically a sequent is true iff, whenever all formulas from Γ are true under some assignment, so is one formula from Δ, i.e. $\bigwedge \Gamma \to \bigvee \Delta$. This especially implies that $\emptyset \longrightarrow \emptyset$ does not hold.

A PK-proof is a sequence of sequents in which every sequent is an initial sequent

$$A \longrightarrow A,\ \bot \longrightarrow,\ \longrightarrow \top,$$

where A is atomic, or is derived from previous sequents by application of the following rules:

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ (\text{Weaken Left}) \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}\ (\text{Weaken Right})$$

$$\frac{\Gamma_1, A_1, A_2, \Gamma_2 \longrightarrow \Delta}{\Gamma_1, A_2, A_1, \Gamma_2 \longrightarrow \Delta}\ (\text{Exchange Left}) \qquad \frac{\Gamma \longrightarrow \Delta_1, A_1, A_2, \Delta_2}{\Gamma \longrightarrow \Delta_1, A_2, A_1, \Delta_2}\ (\text{Exchange Right})$$

$$\frac{\Gamma, A, A \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ (\text{Contract Left}) \qquad \frac{\Gamma \longrightarrow A, A, \Delta}{\Gamma \longrightarrow A, \Delta}\ (\text{Contract Right})$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma, \neg A \longrightarrow \Delta}\ (\neg\ \text{Left}) \qquad \frac{\Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \neg A, \Delta}\ (\neg\ \text{Right})$$

$$\frac{A_1, \Gamma \longrightarrow \Delta}{A_1 \wedge A_2, \Gamma \longrightarrow \Delta}\ (\wedge\ \text{Left 1}) \qquad \frac{\Gamma \longrightarrow A_1, \Delta \qquad \Gamma \longrightarrow A_2, \Delta}{\Gamma \longrightarrow A_1 \wedge A_2, \Delta}\ (\wedge\ \text{Right 1})$$

$$\frac{A_1, \Gamma \longrightarrow \Delta}{A_2 \wedge A_1, \Gamma \longrightarrow \Delta}\ (\wedge\ \text{Left 2})$$

$$\frac{A_1, \Gamma \longrightarrow \Delta \qquad A_2, \Gamma \longrightarrow \Delta}{A_1 \vee A_2 \Gamma \longrightarrow \Delta}\ (\vee\ \text{Left 1}) \qquad \frac{\Gamma \longrightarrow A_1, \Delta}{\Gamma \longrightarrow A_1 \vee A_2 \Delta}\ (\vee\ \text{Right 1})$$

$$\frac{\Gamma \longrightarrow A_1, \Delta}{\Gamma \longrightarrow A_2 \vee A_1 \Delta}\ (\vee\ \text{Right 2})$$

$$\frac{\Gamma \longrightarrow A, \Delta \qquad \Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}\ (\text{Cut})$$

The new formula introduced in a rule is the *principal formula*, the formulas it is inferred from are the *minor formulas*. All other formulas are referred to *side formulas*.

Theorem C.2. *PK is complete. That is, whenever $\Gamma \longrightarrow \Delta$ is true in the semantic sense given above, there is a PK proof of that sequent.*

Resolution

Another important calculus is Resolution. It was introduced by Blake [16] and subsequently developed by Davis and Putnam [32] and Robinson [84]. It is essentially a system for refuting formulas in conjunctive normal form, but can be perceived as a propositional calculus, since there is a canonical way of turning any given formula into an equivalent one in disjunctive normal form. Such a formula is a tautology iff its negation (which can be transformed into a CNF) is refutable.

A propositional formula φ is a k-CNF iff

$$\varphi = \bigwedge_i \bigvee_{j<k} \ell_{i,j},$$

where every $\ell_{i,j}$ is a *literal*, i.e. a variable or a constant, or their negation. A propositional formula φ is a k-DNF iff

$$\varphi = \bigvee_i \bigwedge_{j<k} \ell_{i,j}.$$

We say that a formula is in disjunctive or conjunctive normal form, iff, for some k, it is a k-DNF or k-CNF, respectively. We call $\bigvee_{j<k} \ell_{i,j}$ a *clause* and represent it as the set $\{\ell_{i,0}, \ldots, \ell_{i,k-1}\}$ of its literals.

Resolution consists of only one rule

$$\frac{C_1 \cup \{x\} \qquad C_2 \cup \{\bar{x}\}}{C_1 \cup C_2} \text{ (Res)}$$

where the C_i do not contain x or $\bar{x}$.

A derivation of a clause C from a set of clauses $\mathcal{C}$ is a sequence $A_1, \ldots, A_k = C$, where each A_i is a clause of $\mathcal{C}$ or derived by applying the resolution rule to A_{k_1} and A_{k_2}, for $k_1, k_2 < i$.

A CNF is refuted, iff the empty clause is derivable from its set of clauses.

When we talk about the length of a proof in Resolution we will usually only be interested in proofs of formulas in DNF, as there is no efficient way of turning an arbitrary formula into an equivalent DNF.

If a formula φ is derivable in Resolution we denote that by the symbol $\vdash^{Res} \varphi$.

Theorem C.3. *Resolution, if perceived as above, is a complete calculus.*

C.1.2 First-Order Logic

While in the previous section we were concerned with the relation between statements, we will now turn our focus to the very structure of these statements. That is, we will explore how mathematical statements are built up and try to give a calculus that allows to deduce true statements without referring to the actual mathematical world for proof.

As before we build our formulas inductively from variables, connectives such as $\neg$ and $\vee$ and brackets. Additionally we also allow quantifiers that range over variables, $\exists$ standing for "it exists" and $\forall$ denoting "for all" and an equality symbol $=$. We call these symbols the logical symbols and will now turn to the non-logical ones. Since we want to talk about mathematical structures we also have to add symbols depending on the class of objects we want to talk about.

A *signature* $\sigma =$ is a set of symbols $\{c_1, \ldots, c_m, f_1, \ldots, f_n, R_1, \ldots, R_k\}$ together with a function ar_σ, that defines the arity of each symbol. A signature is relational if $m = n = 0$, that is, if it only contains relation symbols. By convention, we will always include $=$ and interpret it as true identity.

Given a signature $\sigma = (\{c_1, \ldots, c_m, f_1, \ldots, f_n, R_1, \ldots, R_k\}, ar)$, a σ-structure is a set

$$\mathfrak{M} = \{M, c_1^{\mathfrak{M}}, \ldots, c_m^{\mathfrak{M}}, f_1^{\mathfrak{M}}, \ldots, f_n^{\mathfrak{M}}, R_1^{\mathfrak{M}}, \ldots, R_k^{\mathfrak{M}}\},$$

where M is the underlying non-empty set of the structure, called the *universe*, and the $c_i^{\mathfrak{M}}, f_i^{\mathfrak{M}}, R_i^{\mathfrak{M}}$ are the *interpretations* of the constants, functions and relations from the signature. I.e. they are actual objects in our realm of discourse and have names in the signature.

A first-order term is defined inductively as follows.

- Constants and variables are terms.
- If f is a m-ary function symbol and $t_1, \ldots, t_m$ are terms, then so is $f(t_1, \ldots, t_m)$.

A *first-order formula* is defined inductively as

- If t_1, t_2 are terms, then $t_1 = t_2$ is a formula.
- If R is a m-ary relation symbol and $t_1, \ldots, t_m$ are terms, then $R(t_1, \ldots, t_m)$ is a formula.
- If φ_1, φ_2 are formulas, then so are $(\neg\varphi_1)$, $(\varphi_1 \vee \varphi_2)$ and $(\varphi_1 \wedge \varphi_2)$.
- If φ is a formula, so are $(\exists x\varphi)$ and $(\forall x\varphi)$.

Formulas of the first two types are called *atomic*. The set of all formulas in a given signature is called a *language*. As before we use the $\rightarrow$ and $\leftrightarrow$ as abbreviations. A variable is *free* in a formula if it is not in the scope of a quantifier. A formula is a *sentence* if it does not contain free variables. A *theory* is a set of sentences in the same signature (i.e. a subset of the appropriate language). For brevity we will often identify the notion of signature and language, as the latter is a straight-forward construction of the former. As before we will omit unnecessary brackets.

Given a σ-term t with variables $\bar{x}$, a σ-structure M and a sequence of elements $\bar{a}$ of M, we let $t^{\mathfrak{M}}$ be the interpretation of the term t in $\mathfrak{M}$, where a_i is substituted for x_i.

We can now give a definition of what it means that a sentence is true in a mathematical structure. Let σ be a signature and $\mathfrak{M}$ a σ-structure. Then we say that a sentence φ *holds* in $\mathfrak{M}$ (in symbols $\mathfrak{M} \vDash \varphi$) according to the following definition.

Definition C.4. Let σ be a signature and $\mathfrak{M}$ a σ-structure. Let $\varphi(\bar{x})$ be a σ-formula with all free variables displayed. Let $\bar{a} \in M$. Then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff

- If φ is $t_1 = t_2$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff $t_1^{\mathfrak{M}}(\bar{a}) = t_2^{\mathfrak{M}}(\bar{a})$.
- If φ is $R(t_1, \ldots, t_{ar(R)})$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff $R^{\mathfrak{M}}(t_1\mathfrak{M}(\bar{a}), \ldots, t_{ar(R)}^{\mathfrak{M}}(\bar{a}))$.
- If φ is $\neg\psi$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff $\mathfrak{M} \nvDash \psi(\bar{a})$.
- If φ is $\psi_1 \vee \psi_2$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff $\mathfrak{M} \vDash \psi_1(\bar{a})$ or $\mathfrak{M} \vDash \psi_2(\bar{a})$.
- If φ is $\psi_1 \wedge \psi_2$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff $\mathfrak{M} \vDash \psi_1(\bar{a})$ and $\mathfrak{M} \vDash \psi_2(\bar{a})$.
- If φ is $\exists x\psi(\bar{x}, x)$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff there exists $a \in M$ s.t. $\mathfrak{M} \vDash \psi(\bar{a}, a)$.
- If φ is $\forall x\psi(\bar{x}, x)$, then $\mathfrak{M} \vDash \varphi(\bar{a})$ iff for all $a \in M$, $\mathfrak{M} \vDash \psi(\bar{a}, a)$.

Given a theory T, we say that a structure $\mathfrak{M}$ is a *model* of T, in symbols $\mathfrak{M} \vDash T$, iff every sentence from T holds in $\mathfrak{M}$. A theory is *complete* iff for every sentence φ, either $\varphi \in T$ or $\neg\varphi \in T$. We write $T \vDash \varphi$ iff φ holds in every model of T. We write $\vDash \varphi$ iff φ is *valid*, that is, iff it is true in every σ-structure.

As in the previous section we can try to develop a calculus that captures semantical reasoning with syntactical means. This time, however, we are bound to fail if the underlying theory is consistent but too strong. We will see this in Section C.1.2.

Before we come to what we cannot do syntactically we will see some positive results. To this end we will extend the Frege calculus, given above, to a calculus for first-order logic and briefly sketch some classical results.

Let PC be the Frege calculus above. We will extend PC by the following rules of quantifier introduction:

$$\frac{\varphi \to \psi(x)}{\varphi \to \forall x \psi(x)} \ (\forall \text{ introduction}) \qquad \frac{\psi(x) \to \varphi}{\exists x \psi(x) \to \varphi} \ (\exists \text{ introduction})$$

where x may not occur freely in φ. We will also add the following two axiom schemes

$$\varphi(t) \to \exists x \varphi(x) \text{ and } \forall x \varphi x \to \varphi(t)$$

and, for every function symbol f and relation symbol R (including $=$), the following axioms for equality

$$\forall x . x = x$$

$$\forall \bar{x} \forall \bar{y}((x_1 = y_1 \wedge \cdots \wedge x_{ar(f)} = y_{ar(f)}) \to f(\bar{x}) = f(\bar{y}))$$

$$\forall \bar{x} \forall \bar{y}((x_1 = y_1 \wedge \cdots \wedge x_{ar(R)} = y_{ar(R)}) \wedge R(\bar{x}) \to R(\bar{y})).$$

Additionally for $=$ we add

$$a = b \leftrightarrow b = a \text{ and } a = b \wedge b = c \to a = c.$$

We will call this calculus FC and symbolize that a formula φ is derivable in FC by writing $\vdash^{\mathsf{FC}} \varphi$.

Proposition C.5. The system FC is sound and complete. That is, for any set of formulas T, if $T \vDash \varphi$ then $T \vdash^{\mathsf{FC}} \varphi$ and for any set of sentences, if $T \vdash^{\mathsf{FC}} \varphi$ then $T \vDash \varphi$.

A theory T is *consistent* w.r.t. a calculus iff there is a formula that is not derivable from T. We will conclude this overview by giving a short overview of some important properties of First-Order Logic.

Important Theorems for First-Order Logic

In this section we will sketch a few important theorems of first-order logic. This list is by no means complete, but is meant to give a frame to First-Order Logic.

The Compactness Theorem incorporates the intended property that the calculi we have designed are finitistic in the sense that every proof has a finite length.

Theorem C.6 (Compactness Theorem)**.** *A formula φ is provable in FC from a set of formulas Φ iff there exists a finite subset $\Phi_{fin} \subseteq \Phi$, such that φ is already provable from Φ_{fin}.*

From a (first-order) logical perspective, the notion of size of an arbitrary model is not well definable, when we take a look at infinite structures. This is exemplified in the Löwenheim-Skolem Theorem.

Theorem C.7 (Löwenheim-Skolem)**.** *Let L be some first-order language and let Φ be a set of L-formulas that is satisfiable over an infinite universe, with $\kappa = |L|$. Then there is an L-structure $\mathfrak{M}$ of cardinality $\max(\kappa, \aleph_0)$ that satisfies Φ.*

We now turn to the theorems that obliterated Hilbert's vision of being able to turn every ideal proof into a finitistic one. Gödel's Incompleteness Theorems [45] show that there is no formal system that admits on the one hand a decidable notion of provability and on the other hand is complete in the sense that it can prove every Π_1 statement that holds in the natural numbers.

Numerous books have been written about the Incompleteness Theorems, so we will only state the theorems. We refer a more interested reader to the literature, for example [8], [22] or [80].

We will first give Gödel's original version of his first theorem and then Rosser's version, which does not need the additional assumption of ω-consistency. A theory T is called *ω-consistent* iff it is consistent in a way with the natural numbers. That is, if an existential statement $\exists\varphi(x)$ is provable from T, then there must be a numeral $\underline{n}$ such that T does not prove $\neg\varphi(\underline{n})$.

Theorem C.8 (Gödel's First Incompleteness Theorem)**.** *Let T be a consistent, recursive theory extending Robinson Arithmetic. Then there is a true sentence φ, such that φ is not provable from T. Moreover, if T is ω consistent, then $\neg\varphi$ is also not provable from T.*

Here, the formula φ is explicitly constructed from T and basically asserts its own unprovability from T, i.e. it is an encoding of the statement "I am not provable in T". Rosser [85] subsequently weakened the assumptions of Gödel's Theorem.

Theorem C.9 (Rosser's Theorem)**.** *There is no consistent, recursive and complete theory extending Robinson Arithmetic.*

Soon after publishing his First Incompleteness Theorem, Gödel became aware of the existence of another example example of a statement that is not provable in T.

Theorem C.10 (Gödel's Second Incompleteness Theorem)**.** *There is no consistent, recursive theory extending Robinson Arithmetic that can prove its own consistency.*

The proof of Theorem C.10 actually shows that the consistency statement is equivalent to the sentence from Theorem C.8 asserting its own unprovability.

C.2 Complexity Theory

Gödel's Incompleteness Theorems imply that there are problems that cannot be solved algorithmically. In Complexity Theory we are concerned with those problems that can be. Even if problems are solvable they differ in various aspects,

such as time and space needed for the computation. To make these notions precise we first need to define what our model of computation should be. We will use a *Turing machine* as our principal model of computation.

Definition C.11. A *deterministic Turing machine* is defined by a tuple (K, Σ, δ, s), where K is a set of states containing the initial state s, Σ a set of symbols, containing $\triangleright$ (the "first" symbol) and $\sqcup$ (the "blank" symbol), and $\delta : K \times \Sigma \longrightarrow K \cup \{"halt", "yes", "no"\} \times \Sigma \times \{\leftarrow, \rightarrow, \downarrow\}$ the transition function. The state "*halt*" is called the *halting state*, the state "*yes*" the *accepting state* and the state "*no*" the *rejecting state*.

We view a Turing machine as a means of computation in the following sense. The machine works on a to the right infinite *work tape* consisting of *cells*, each of which can be filled with symbols from Σ. It has one *read/write head* which points to a cell. Initially the tape contains only the input $x \in \Sigma^*$ on its leftmost cells and $\sqcup$ on the others. The read/write head points to the leftmost cell and the machine is in its initial state s. A Turing machine computation is a sequence of applications of the transition function until the machine is in one of the states "*halt*", "*yes*" or "*no*". The transition function is applied as follows. If the read/write head points to cell i, the machine is in state t and the cell contains symbol $y \in \Sigma$, then the machine looks at $\delta(t, y) = (t', y', dir)$ and replaces y on cell i by y', moves in direction dir, that is if $dir = \leftarrow$ it points to cell $i \dot{-} 1$, if $dir = \rightarrow$ it points to cell $i + 1$ and if $dir = \downarrow$ it points to cell i, and then the machine goes into state t'. If the state is not one of "*halt*", "*yes*", "*no*" it continues to do so. We call the number of applications of Δ the *number of steps*. A *configuration* of a Turing machine is a snapshot of its whole tape, state and position of the read/write head during its computation. We can picture the computation of a Turing machine as a sequence of its configurations.

A Turing machine M *accepts* an input $x \in \Sigma^*$, in symbols $M(x) = "yes"$, iff the last state of the computation on input x is "*yes*", it *rejects* x, in symbols $M(x) = "no"$, if the outcome is "*no*".

A Turing machine M *decides* or *computes* a language $L \subset \Sigma^*$, iff for all $x \in \Sigma^*$, $x \in L \Leftrightarrow M(x) = "yes"$ and $x \notin L \Leftrightarrow M(x) = "no"$. We call a function computable if its bit graph is decidable.

Turing machines can be defined to effectively simulate other Turing machines.

Theorem C.12. *There exists a* universal Turing machine U. *That is, on input* $\langle x, \tilde{M} \rangle$, *where* $\tilde{M}$ *is an encoding of the description of Turing machine* M, U *simulates the computation of* M *in the sense that* $M(x) = "yes" \Leftrightarrow U(\langle x, \tilde{M} \rangle) = "yes"$, $M(x) = "no" \Leftrightarrow U(\langle x, \tilde{M} \rangle) = "no"$ *and* $M(x) = "halt" \Leftrightarrow U(\langle x, \tilde{M} \rangle) = "halt"$. *Moreover there exists a polynomial* p, *such that the number of steps* U *needs for these computations is bounded by* p *of the number of steps* M *needs.*

A Turing machine M *halts* on input x, iff it reaches one of the states "*halt*", "*yes*" or "*no*" in its computation. The famous *halting problem* asks, whether there is a Turing machine that computes the language

$$\mathsf{HALT} := \{x \in \{0,1\}^* : x \text{ encodes a Turing machine that halts on every input}\}.$$

The following can be shown via a simple diagonal argument.

Theorem C.13. *There is no Turing machine computing* HALT.

As we have observed so far, Turing machine constitute a robust means of computation. That goes as far as the famous *Church-Turing Thesis*, which states that every language that can be computed by any means can also be computed by a Turing machine.

We will turn now to a more contemporary perception of computations. That is, we will be interested in the number of steps a machine takes in a computation and devise classes which take such things into account. We call the number of steps a Turing machine needs to compute an input of length n its *time*, measured as a function $f(n)$. The *space* needed is the maximal number of a cell the read/write head points to during a computation. We say that a TM M has *time complexity* f, if it uses at most $f(n)$ many steps on inputs of length n. The same goes for *space complexity* with respect to the number of cells needed.

When interested in size or time, suddenly new computing paradigms appear which did not make a difference when one only focusses on computability. The one we will be interested in the most is *nondeterminism*. To define this we have to revisit the definition of a (deterministic) Turing machine given above. Observe that the transition function above is what it is: a function. If we relax the definition, we could also allow for the transition being given by a relation, i.e. allowing more than one possible value. To still make sense of the notion of computation, we have to give a way of deciding whether a Turing machine accepts or rejects. To this end we take a look at the tree of all possible computations on input x. We say that a nondeterministic Turing machine accepts an input x if one branch of computations leads to "*yes*" and it rejects if all leafs are "*no*". The time needed for the computation is the smallest length of a branch leading to "*yes*".

This definition is equivalent to the following heuristic. A decision problem is decidable in nondeterministic time f, iff a witness exists that can be checked in time f by a deterministic Turing machine. We let $\mathsf{Time}(f)$ be the class of all languages that are decided by a deterministic Turing machine in f many steps. Accordingly we call $\mathsf{NTime}(f)$ the class of all languages that are accepted by a nondeterministic Turing machine in f many steps. We define the space classes $\mathsf{Space}(f)$ and $\mathsf{NSpace}(f)$ and the classes with space and time bound $\mathsf{TimeSpace}(f)$ and $\mathsf{NTimeSpace}(f, g)$ respectively. Obviously, for all f, g, $\mathsf{Time}(f) \subseteq \mathsf{NTime}(f)$, $\mathsf{Space}(f) \subseteq \mathsf{NSpace}(f)$ and $\mathsf{TimeSpace}(f, g) \subseteq \mathsf{NTimeSpace}(f, g)$. We generalize these notions to families of functions in the obvious way. We call any class of languages a *complexity class*. Arguably the most famous complexity classes are $\mathsf{P} := \mathsf{Time}(poly)$ and $\mathsf{NP} := \mathsf{NTime}(poly)$, where *poly* is the set of all polynomials. Another interesting class is FP, the class of all functions whose bit graph is in P. The famous question, whether $\mathsf{P} = \mathsf{NP}$ asks, whether there is an efficient way to conduct nondeterministic computations on deterministic machines. Interestingly, the question has been settled affirmatively with respect to space complexity by Savitch in [86].

Theorem C.14 (Savitch)**.** *Let f be any eventually increasing function that grows at least as strong as the logarithm, then* $\mathsf{NSpace}(f(n)) \subseteq \mathsf{Space}((f(n))^2)$.

Another important family of complexity classes are those, whose complement is one of the aforementioned classes. We denote these classes by a co written

in front of them. For example the class of problems, whose complements have polynomial size witnesses that can be verified in deterministic polynomial time is the class coNP. This class is of high importance to us, as it contains an abundance of interesting problems. Namely those, which have a feasible refutation.

A *many-one reduction* from a language $L \subset \{0,1\}^*$ to a language $L' \subset \{0,1\}^*$ is a polynomial time computable function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ such that $x \in L \Leftrightarrow f(x) \in L'$. Iff there is such a reduction from L to L' we denote this by $L \leq_m L'$. If C is any complexity class and $L \in C$ is a language computable in this class, then L is *complete* for C iff for all $L' \in C$, $L' \leq_m L$. Complete problems form a very important subclass of a complexity class as we can relate complexity classes to one another simply by relating complete problems. The notions of reduction and completeness can straightforwardly be generalized to functions from other classes than FP.

As a first example of a complete language we take $3SAT$ the set of all satisfiable 3CNF (in some fixed encoding). It is in NP, because a satisfying assignment is a polynomial size witness that can be efficiently verified. On the other hand, by carefully observing how computations on Turing machines are carried out, we can also see that any such computation can be given as a 3CNF. This was famously first proved by Cook in [26] and independently by Levin in [63].

Theorem C.15. *[Cook-Levin]* $3SAT$ *is* NP*-complete.*

As a straightforward application this implies that, to answer the $\mathsf{P} = \mathsf{NP}$ question, it is sufficient to decide, whether there is a polynomial-time algorithm for deciding whether a 3CNF is satisfiable or not. Unfortunately, it turned out that this question is not easy to answer at all.

Another example for a complete language is TAUT, the class of all propositional tautologies (in some arbitrary, but sensible, fixed encoding), which is coNP-complete. This is an easy corollary from Theorem C.15, as a formula φ is a propositional tautology if and only if its negation $\neg\varphi$ is not satisfiable.

An *oracle Turing machine* is a Turing machine M that has an additional tape assigned to some fixed *oracle* O (i.e. predicate), the *query tape*, and an additional state, the *query state*. At any point when the machine is in the query state it reads what is written on its query tape and, in one step, infers the solution of that query from the oracle assigned to that tape. That is, if the query satisfies the predicate, it will write 1 on the tape, otherwise 0. We denote this machine by M^O.

This notion gives rise to another version of reducibility. We say that a language L *Turing reducible* to a language L' (in symbols $L \leq_T L'$) iff the there exists an oracle Turing machine M with an oracle for L' that decides L. We say that this reduction is an f-time reduction, if M uses at most $f(n)$ many steps for inputs of size n (in symbols $L \leq_T^f L'$). This notion of reducibility is weaker than the notion of many-one reducibility introduced earlier as the following holds.

Proposition C.16. Let $L, L' \subset \Sigma^*$ be two languages and let f be an eventually not too slowly increasing function. Then $L \leq_m^f L' \Rightarrow L \leq_T^f L'$.

We say that a language L is in the class P^{NP} iff there is a deterministic polynomial-time Turing machine M and a many-one NP-complete oracle O, such that M^O decides L. Accordingly for $\mathsf{NP}^{\mathsf{NP}}$ and $\mathsf{coNP}^{\mathsf{NP}}$. We also call these classes

Δ_1^p, Σ_1^p and Π_1^p, respectively. We let $\Delta_0^p := \mathsf{P}$, $\Sigma_0^p := \mathsf{NP}$, $\Pi_0^p := \mathsf{coNP}$ and then define

- $\Delta_{i+1}^p := \mathsf{P}^{\Sigma_i^p}$,
- $\Sigma_{i+1}^p := \mathsf{NP}^{\Sigma_i^p}$ and
- $\Pi_{i+1}^p := \mathsf{coNP}^{\Sigma_i^p}$.

The following relation holds.

Proposition C.17. For all i, $\Delta_i^p \subseteq \Sigma_i^p$, $\Delta_i^p \subseteq \Pi_i^p$, $\Pi_i^p \subseteq \Delta_{i+1}^p$ and $\Sigma_i^p \subseteq \Delta_{i+1}^p$.

We let the polynomial hierarchy $\mathsf{PH} := \bigcup_i \Sigma_i^p$. Thus, the $\mathsf{P} = \mathsf{NP}$ question asks, whether the polynomial hierarchy collapses to its first level. There is a nice correspondence between computability in various levels of the polynomial hierarchy and provability in weak arithmetic, this is well explained in [20] and [57].

C.2.1 Circuit Complexity

We will give an exposition of the circuit classes which are of relevance to us. For a thorough introduction see [90]. Informally speaking a circuit is a very simple means of computation defined by an underlying labeled graph. It is defined as follows.

Definition C.18. A *Boolean Circuit* C is a labeled directed acyclic graph containing sources (nodes with in-degree 0, the *input nodes*) and sinks (nodes with out-degree 0, the *output nodes*). Each node that is not an input node is labeled with either $\wedge$, $\vee$ or $\neg$, where nodes which are labeled as $\neg$ have in-degree at most 1, while the others may have any in-degree. The input nodes and output nodes are ordered, such that we can refer to the ith bit of the input or output. The *size* $|C|$ of a circuit C is the number of its nodes, the depth of a circuit is the length of a longest path in C. A circuit is *layered* if, starting from every input node, any node with the same distance to that node is of the same type (i.e. $\wedge$, $\vee$ or $\neg$). A circuit is *monotone* iff it does not contain any negation. A formula can be perceived as a circuit whose underlying structure is a tree.

When talking about circuits we usually refer to the nodes as *gates* and to the in-degree as *fan-in*.

We can compute the *value* of a circuit as follows.

Definition C.19. Let C be a circuit and I an input of length n given in binary. Then the value of its nodes is defined recursively as follows.

- The value of an input node is the value of the appropriate bit of I.
- If the node is labeled $\neg$, then its value is 1 minus the value of its predecessor.
- If the node is labeled $\vee$, then its value is the maximum of the values of the predecessors.

- If the node is labeled $\wedge$, then its value is the minimum of the values of the predecessors.

We denote the value of the output node(s) of C on input I as $C(I)$.

We can now formally state what we understand as languages computed by circuits.

Definition C.20. Let $(C_i)_{i\in\mathbb{N}}$ be a family of circuits where C_i has exactly i input bits. Then $(C_i)_{i\in\mathbb{N}}$ computes $L \subseteq \{0,1\}^*$ if and only if

$$X \in L \Leftrightarrow C_{|X|}(X) = 1.$$

$(C_i)_{i\in\mathbb{N}}$ computes a function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ if and only if

$$f(X) = Y \Leftrightarrow C_{|X|}(X) = Y.$$

This quite simple model of computation is still very powerful. As we need a different circuit for each input length, we can "compute" non-recursive problems using circuits.

Proposition C.21. Let L be a non-recursive language and let L_{tall} be a non-recursive language defined by $x \in L_{tall}$ iff $|x|$ in binary is in L. Then there is a circuit family C_n that computes L_{tall}.

Proof. For all $i \in \mathbb{N}$ let C_i be such that

$$C_i(X) = \begin{cases} 1 & \text{if } 1^i \in L \\ 0 & \text{else.} \end{cases}$$

Obviously this circuit computes L_{tall}. □

Thus, to make circuits a sensible model for computation we have to be able to construct each circuit in a family from some finite description. This idea is facilitated as follows.

Definition C.22. Let K be any complexity class. A family C_i of circuits is K-uniform iff there exists a K-Turing machine M such that for all i, M computes C_i on input 1^i.

In the case of monotone circuits we can usually assume that they are layered.

Proposition C.23. If C is a monotone circuit, then there exists a layered monotone circuit C' that computes the same and whose size and depth differ from C's by only a constant factor.

We will now define some important circuit classes.

Definition C.24. Let n denote the length of the input.

- AC^0 is the class of all languages that can be computed by L-uniform circuits with polynomial size and constant depth.
- AC^k is the class of all languages that can be computed by L-uniform circuits with polynomial size and depth $O(\log(n)^k)$.

- NC^0 is the class of all languages that can be computed by L-uniform circuits with polynomial size and constant depth, where the fan-in of any gate is at most 2.
- NC^k is the class of all languages that can be computed by L-uniform circuits with polynomial size and depth $O(\log(n)^k)$, where the fan-in of any gate is at most 2.

We define $\mathsf{AC} = \bigcup_k \mathsf{AC}^k$ and $\mathsf{NC} = \bigcup_k \mathsf{NC}^k$.

These classes relate to each other as follows.

Proposition C.25. For any k

$$\mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{NC}^{k+1}.$$

Therefore $\mathsf{NC} = \mathsf{AC}$.

One can, in principle, define the gates in a circuit by any boolean function. This leads to various stronger circuit classes, the most important of which is probably TC, which is defined using threshold gates.

Definition C.26. For any i the *threshold function* $Th_i : \{0,1\}^* \longrightarrow \{0,1\}$ is defined as

$$x \mapsto \begin{cases} 1 & \text{if the number of 1 bits in } x \text{ is at least } i \\ 0 & \text{else} \end{cases}.$$

Using this function we can define the circuit class TC.

Definition C.27. For any k let TC^k be the class of all languages that can be computed by L-uniform circuits using $\wedge, \vee, \neg$ and Th_i gates and having polynomial size and depth $O(\log(n)^k)$. We let $\mathsf{TC} := \bigcup_k \mathsf{TC}^k$.

As we will prove now, threshold functions can be computed in AC^1. Therefore the class TC equals AC.

Proposition C.28. $\mathsf{TC}^k \subseteq \mathsf{AC}^{k+1}$.

Proof. The threshold function $Th_i(x_1 \dots x_m)$ can be computed inductively by

$$Th_i(x_1 \dots x_m) = \bigvee_{k=0}^{i} (Th_k(x_1 \dots x_{\lfloor \frac{m}{2} \rfloor}) \wedge Th_{i-k}(x_1 \dots x_{\lfloor \frac{m}{2} \rfloor})).$$

□

However, the classes differ provably on their lower levels. To see this we will first introduce a boolean function and a corresponding circuit class.

Definition C.29. Let $MOD_i(x_1 \dots x_m) \equiv \sum_{k=1}^{m} x_k \mod i = 0$. For any k let $\mathsf{AC}^k[i]$ be the class of all languages that can be computed by L-uniform circuits using $\wedge, \vee, \neg$ and MOD_i gates and having polynomial size and depth $O(\log(n)^k)$.

A famous result of Smolensky is the following theorem [88].

Theorem C.30 ([88]). *Let $gcd(p, r) = 1$. Then $MOD_r \notin \mathsf{AC}^0[p]$.*

But for any i, MOD_i can be computed using $O(i)$ many threshold gates by

$$MOD_i(x_1 \dots x_m) \equiv \bigvee_{k|i} (Th_k(x_1 \dots x_m) \wedge \neg Th_{k+1}(x_1 \dots x_m)).$$

Therefore we have

$$\mathsf{AC}^0 \subsetneq \mathsf{AC}^0[p] \subsetneq \mathsf{TC}^0 \subseteq \mathsf{NC}^1.$$