Combinatorics in Bounded Arithmetic
(Ph.D. Dissertation)

Kerry Ojakian

June 23 2004

Committee:
Jeremy Avigad (Advisor)
James Cummings
Ramamoorthi Ravi
Rick Statman

Abstract

A basic aim of logic is to consider what axioms are used in proving various theorems of mathematics. This thesis will be concerned with such issues applied to a particular area of mathematics: combinatorics. We will consider two widely known groups of proof methods in combinatorics, namely, probabilistic methods and methods using linear algebra. We will consider certain applications of such methods, both of which are significant to Ramsey theory. The systems we choose to work in are various theories of bounded arithmetic.

For the probabilistic method, the key point is that we use the weak pigeonhole principle to simulate the probabilistic reasoning. We formalize various applications of the ordinary probabilistic method and linearity of expectations, making partial progress on the Local Lemma. In the case of linearity of expectations, we show how to eliminate the weak pigeonhole principle by simulating the derandomization technique of "conditional probabilities."

We consider linear algebra methods applied to various set system theorems. We formalize some theorems using a linear algebra principle as an extra axiom. We also show how weaker results can be attained by giving alternative proofs that avoid linear algebra, and thus also avoid the extra axiom.

We formalize upper and lower Ramsey bounds. For the lower bounds, both the probabilistic methods and the linear algebra methods are used. We provide a stratification of the various Ramsey lower bounds, showing that stronger bounds can be proved in stronger theories.

A natural question is whether or not the axioms used are necessary. We provide "reversals" in a few cases, showing that the principle used to prove the theorem is in fact a consequence of the theorem (over some base theory). Thus this work can be seen as a (humble) beginning in the direction of developing the Reverse Mathematics of finite combinatorics.

Acknowledgments

I would like to thank my parents for understanding my work in their own way. Thanks to numerous friends for providing diversion from the work. And of course thanks to my advisor Jeremy Avigad for his patience and guidance. Thanks to the rest of my committee: James Cummings, Ramamoorthi Ravi, and Rick Statman. A number of other mathematicians have helped in various ways: Steve Cook, Paola D'Aquino, Jan Krajíček, Pavel Pudlák, and Steve Simpson.

Contents

Chapter 1

Introduction

A basic aim of logic is to consider what axioms are used in proving various theorems of mathematics. This thesis will be concerned with such issues applied to a particular area of mathematics: combinatorics. We will consider two widely known groups of proof methods in combinatorics, namely, probabilistic methods and methods using linear algebra. We will consider certain applications of such methods, both of which are significant to Ramsey theory. The systems we choose to work in are various theories of **bounded arithmetic**.

Others have considered formalizing various aspects of mathematics in the context of bounded arithmetic. Paris, Wilkie and Woods [39] did some number theory, in particular, showing the infinitude of the primes. D'Aquino and Macintyre have formalized a number of theorems of number theory ([14], [15], [16], [17]). Soltys and Cook [43] have considered aspects of linear algebra, developing systems for feasible reasoning about matrices. Pudlák [40] formalized Ramsey's theorem. Pudlák's work inspired much of this work, with Ramsey theory motivating much of our formalization.

The term "bounded arithmetic," originally applied to Parikh's $I\Delta_0$ [38], is used of a number of different theories. However the essential commonality is that such a theory is axiomatized by a set of bounded formulae. A consequence of this is that such a theory can not prove the totality of the exponential function (assuming the language is sufficiently limited). This limitation is especially interesting for finite combinatorics because much of it can be formalized without a thought in a theory of arithmetic that has exponentiation. Thus trying to formalizing combinatorics within some theory of bounded arithmetic is like asking how the proofs and notions must change in the absence of exponentiation.

One of the most significant properties of theories of bounded arithmetic is their close connection to computational complexity theory. Cook [11] originally developed the equational theory called PV with the goal of capturing polynomial time reasoning. Buss [9] developed a first-order arithmetic

version of this, called S^1_2. The provably total functions given by definitions of a certain complexity are exactly the polynomial time functions (this will be discussed in detail in chapter 2). He extended S^1_2 to an entire hierarchy of bounded theories called S_2, which correspond to the polynomial time hierarchy. In an analogous way, the original theory $I\Delta_0$ corresponds to the linear time hierarchy, though we will not discuss this. We will be concerned with formalizing proofs in the vicinity of the theory S^1_2, considered one of the most fundamental theories of bounded arithmetic, due to its connection to one of the most studied complexity classes.

After discussing some of the details of first-order bounded arithmetic (in chapter 2), in chapter 3 we consider some well-known probabilistic method proofs. These are non-constructive proofs which argue that certain objects exist without constructing the object. As would be expected, such proofs will require a little more than S^1_2. Even with a little extra, there is a problem dealing with the probability spaces or carrying out all the counting in the proofs. The basic idea of this chapter is to simulate this counting argument using the **weak pigeonhole principle**. In some cases, we can avoid the use of the weak pigeonhole principle, by formalizing an algorithm.

One of the applications of the probabilistic methods we formalize in chapter 3 is a non-constructive Ramsey lower bound; the proof shows that a certain coloring exists without exhibiting it explicitly. We consider constructive Ramsey lower bounds in the chapter on Ramsey theory, chapter 6. One of the main constructive lower bounds we consider is the well-known construction of Frankl and Wilson [25]. They use a construction based on **set systems**, which let the vertices be certain sets, and color the edges according to the intersection size of these sets. The key to their proof is to apply linear algebra methods to such set systems. Thus in chapter 5 we develop the necessary linear algebra to formalize this argument and other work on set systems.

Though one of the main motivations of chapter 5 is its application to Ramsey theory, we also consider set system theorems for their own sake. For this work, we will use second-order systems of bounded arithmetic, working in between the theory V^0 and the strictly stronger theory V^1, developed by Zambella [48], but based on Buss [9]; we discuss second-order bounded arithmetic in chapter 4. The second-order theory V^1 is isomorphic to the first-order theory S^1_2, so we essentially use these second-order systems as a convenient way to work below our central theory S^1_2 (see the beginning of chapter 4 for a picture of how the various theories hang together). The typical set system argument (i.e. the ones from [5]) use linear algebra and so we formalize them in the theory V^0 plus some principles, including a special linear algebra principle (this mostly takes place in V^1). We have found that we can give alternative proofs which avoid linear algebra but have worse bounds. Thus we have a trade off between the strength of the axioms used and the strength of the bound in the theorem. We refer to this phenomenon as **the proof-strength vs. bound-strength trade-off.**

Chapter 6 on Ramsey theory has already been discussed in so far as how it fits in with other aspects of the thesis. In this chapter we will also look at some Ramsey upper bounds. Pudlák [40] originally proved such a bound in bounded arithmetic. Working in the same theory, we improve his

bounds. We will also look at the idea of **reversals**, showing that some Ramsey principles imply the key principles used to prove them. We will prove a reversal in the context of the set system theorems of chapter 5.

The basic motivation for this work can be seen as analogous to that of **Reverse Mathematics** and the related work on subsystems of second order arithmetic (see Simpson's book [42]). In that work they formalize ordinary mathematics, taking RCA_0 as their "base theory," meaning that they always assume at least that much proof-strength, adding axioms as needed. The idea is of course to add the weakest axioms necessary to prove the theorem at hand; this can be made precise by proving a reversal over the base theory, showing that the theorem is in fact equivalent to the extra axiom used to prove it. Though RCA_0 is weaker than Peano Arithmetic, it is strong enough to assert the existence of recursive sets. For lots of finite combinatorics such a base theory is much stronger than necessary, thus to develop a similar program of Reverse Mathematics for finite combinatorics requires a much weaker context. Bounded Arithmetic is a very convenient context for such a study. The proofs we provide do not seem to be using much more proof strength than required, and we in fact have reversals in some cases. We discuss some more details along these lines in the conclusion.

Notation

We give some widely used notation which will be discussed, but is gathered here for easy reference.

- Let $\mathbb{N}$ be the set of natural numbers $\{0, 1, 2, \ldots\}$.
- For functions and relations on $\mathbb{N}$ and subsets of $\mathbb{N}$, we use the **boldface** font (e.g. function $\mathbf{f}$ and relation $\mathbf{R}$).
- For formulae we use $\mathsf{this\ font}$.
- For function and relation symbols in some language we use `this font`.
- For a natural number n, $[n]$ is the set $\{0, 1, \ldots n-1\}$.
- $\vec{x}$ or $\vec{X}$ is a finite list of variables in some language.
- $\mathsf{A} := \mathsf{B}$ means that A is *defined* to be equal to expression B, using this notation to distinguish from equality testing.
- $\log x$ means $\lfloor \log_2 x \rfloor$ unless otherwise mentioned.
- $\mathsf{pow}(x, y)$ is x^y.
- $\|X\|$ is the size of set X.

- $For X \subseteq Y$, $\overline{X}$ is the complement of X in Y; Y will be indicated by context.
- A **multifunction** on some domain D, is a binary relation R such that for all $x \in D$, there is a y such that $R(x, y)$.

Chapter 2

First-Order Bounded Arithmetic

Peano Arithmetic (PA) is one of the most well-known and well-studied logical theories. It is intended to be a theory of the natural numbers $\mathbb{N}$, with the usual arithmetic operations such as addition and multiplication. This theory consists of a finite number of defining axioms for the basic operations, along with an infinite schema of induction axioms. A typical choice of language for PA consists of the set $\{+, *, 0, 1, \leq, =\}$, these symbols having their usual intended interpretations on $\mathbb{N}$. In general when we refer to a **language** we mean some set of symbols (function, relation, and constant symbols) beyond the usual logical symbols of first-order logic; terms and formulae are built up in the usual way. We have the usual logical symbols, though we use "$\Rightarrow$" and "$\Leftrightarrow$" for the implication and equivalence symbols, respectively. By convention, "$\Rightarrow$" and "$\Leftrightarrow$" bind most weakly. By a **theory** we simply mean a set of formulae built up from some language (we do not require a theory to be deductively closed). We will always be dealing with theories of arithmetic that are weaker than PA. The various strengths of the theories we work with will essentially depend on the strength of their induction axioms.

Definition 2.0.1 *Given a formula ψ, let $\psi-$IND be:*

$$\psi(0) \wedge \forall x \ (\psi(x) \Rightarrow \psi(x+1)) \Rightarrow \forall x \ \psi(x).$$

PA has $\psi-$IND for any first-order formula ψ. The various theories of bounded arithmetic are basically PA, with $\psi-$IND restricted to ψ which are **bounded formulae** (to be defined). Parikh [38] originally introduced bounded arithmetic in 1971, motivated by the idea of trying to capture feasible reasoning; his theory is called $\mathrm{I}\Delta_0$. One of the desired limitations of this theory is that the exponentiation function cannot be defined, so given a number n, we do not know that 2^n exists. However for a number of purposes $\mathrm{I}\Delta_0$ is too limited. In PA, all sorts of sequences can be coded by numbers, while in $\mathrm{I}\Delta_0$ only sequences of some standard length can be coded (by a **standard** natural number we mean an actual natural number in the set $\mathbb{N}$, as opposed to an element of some

non-standard model of arithmetic). This limitation interferes with developing meta-mathematics within the system, as well as (apparently) hampering our ability to formalize certain mathematics. A compromise is to strengthen $I\Delta_0$ with an axiom that states that for any natural number n, $n^{\lfloor \log n \rfloor}$ exists; this axiom is called Ω_1. The theory $I\Delta_0 + \Omega_1$ is still too weak to define the exponentiation function, but it is strong enough to more easily formalize mathematics; in particular we can now work with "short" sequences whose length is $< (\log n)^k$ for a standard k. In 1986 Buss [9] developed S_2, a theory which is essentially equivalent to $I\Delta_0 + \Omega_1$. We will discuss $I\Delta_0$ and S_2. Then we discuss some issues relating to both theories.

We note some technical conventions used here and throughout this thesis. We typically refer to the formulae in a theory as **axioms**. When these axioms have free variables we mean the universal closure of the axioms. It may be helpful to keep in mind that all the theories will be defined by specifying a finite set of basic defining axioms for the symbols of the language (denoted BASIC with some superscript), along with a schema of induction axioms for formulae of some specified complexity. For a set of formulae Φ, Φ−IND refers to the set of formulae $\{\psi\text{−IND} \mid \psi \in \Phi\}$. A finite list of variables is abbreviated $\vec{x}$ using some letter x. Unless otherwise stated, it is always assumed that the free variables of a formula $\psi(\vec{x})$ are contained in $\vec{x}$. However we will sometimes state that a formula may contain unmentioned free variables; we use the name **parameters** loosely to describe variables that may not be explicitly included in a formula description. If we exhibit a formula with a semi-colon as in $\psi(\vec{x};\vec{a})$, we mean that the $\vec{a}$ are parameters; we may leave off this list of parameters just writing $\psi(\vec{x})$. Intuitively, parameters are fixed objects with some property as opposed to arbitrary variables that we quantify over. Since we always deal with theories of arithmetic, a number is a natural number, unless otherwise stated; we let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers. For a number n, by $[n]$ we mean the set $\{0, 1, \ldots, n-1\}$. When we refer to a number n as the domain or range of a function we mean the set $[n]$. By $[a, b]$, for $a \leq b$, we mean the set $\{a, a+1, \ldots, b\}$; the intervals (a, b), $(a, b]$, and $[a, b)$ are defined similarly. We will sometimes write $\mathsf{A} := \mathsf{B}$ as a short way to *define* A to be equal to B.

2.1 The Theory $I\Delta_0$

We discuss some aspects of $I\Delta_0$; a good reference for more detail is [27, chapter 5]. The language of the theory is $\{+, *, 0, 1, \leq, =\}$. We have the following basic axioms.

Definition 2.1.1 *Let* BASIC^0 *be:*

1. $x + 1 \neq 0$
2. $x + 1 = y + 1 \Rightarrow x = y$
3. $x + 0 = 0$

4. $x + (y + 1) = (x + y) + 1$
5. $x * 0 = 0$
6. $x * (y + 1) = (x * y) + x$
7. $x \leq x + y$
8. $(x \leq y \wedge y \leq x) \Rightarrow x = y$
9. $0 + 1 = 1$

We will introduce the induction axiom for various classes of bounded formulae, defined using the following restricted notions of quantification.

Definition 2.1.2 *For a term t and a formula ψ:*

- *Let $\exists x < t\ \psi$ abbreviate: $\exists x\ (x < t \wedge \psi)$.*
- *Let $\forall x < t\ \psi$ abbreviate: $\forall x\ (x < t \Rightarrow \psi)$.*

By a **first-order bounded quantifier** we mean a quantifier of one of the two above types. We may leave off the qualification of "first-order," when the meaning is clear. A formula that only has bounded quantifiers is called a **bounded formula**; the set of such formulae is denoted Δ_0. We can now define Parikh's original notion of bounded arithmetic.

Definition 2.1.3 $\mathrm{I}\Delta_0$ *is the theory consisting of* BASIC^0 *and* Δ_0*−IND.*

As mentioned, a key aspect of $\mathrm{I}\Delta_0$ is that the exponentiation function cannot be defined. This will follow from Parikh's theorem which states that for any function definable with a Δ_0 formula, we can bound the value of the function by a term in the language. This theorem in fact holds for any **bounded theory**, this being a theory that can be axiomatized by bounded formulae (allowing free variables). Notice that $\mathrm{I}\Delta_0$ is a bounded theory because the induction axioms can be restated as bounded formulae. Now we can state Parikh's theorem in a general way referring to any bounded theory.

Theorem 2.1.4 *(Parikh's Theorem [38]) Suppose* T *is a first-order bounded theory containing* $\mathrm{I}\Delta_0$ *and $\psi(\vec{x}, y)$ is a Δ_0 formula such that* T *proves $\forall\vec{x}\exists y\ \psi(\vec{x}, y)$. Then there is a term t, not containing y, such that* T *proves $\forall\vec{x}\exists y < t\ \psi(\vec{x}, y)$.*

One of the main applications of Parikh's theorem is to show that there is no Δ_0 definition of the **exponentiation function** in a bounded theory T. Let $\mathbf{exp}(x, y, z)$ be the relation that holds if $x^y = z$. Here and throughout, we use **boldface** to refer to actual relations on $\mathbb{N} = \{0, 1, 2, \ldots\}$ or subsets of $\mathbb{N}$, distinguishing them from formulae in some language. If there were a Δ_0 definition of the exponentiation function in T, that would mean that that T could prove that $\forall x, y\, \exists z\, \mathsf{exp}(x, y, z)$, where exp is a Δ_0 definition of the relation **exp**. The problem would be that by Parikh's theorem, T would then be able to prove that $\forall x, y\, \exists z < t(x, y)\, \mathsf{exp}(x, y, z)$, for some term t in the language; this is false, because terms in this language are polynomials, which grow slower than the exponential function. However, a non-obvious fact (by Paris, but discussed in [20]) is that the **exponentiation relation** $\mathbf{exp}(x, y, z)$ is definable by a Δ_0 formula exp, and $\mathrm{I}\Delta_0$ can prove the basic properties of this formula (see [27, p.299]). We will often re-visit this useful relation. *Throughout this thesis, if we say that a function or relation on* $\mathbb{N}$*, named say "***example***," is definable by some formula, then we will henceforth use "*example*" to refer to this particular definition.*

It is common in any theory to add a number of functions and relations that make life easier without increasing the power of the theory, that is, to add them conservatively. It is well known that we can simply add definable functions and relations.

Definition 2.1.5 *Let* Φ *be some class of formulae.*

- *A relation* $\mathbf{R}(\vec{x})$ *is* Φ ***definable*** *if there is a formula* $\psi(\vec{x})$ *in* Φ *such that* $\mathbf{R}(\vec{x})$ *holds iff* $\psi(\vec{x})$ *holds in* $\mathbb{N}$*.*
- *A function* $\mathbf{f}(\vec{x})$ *is a* Φ ***definable function in theory T*** *if there is a formula* $\psi(\vec{x}, y)$ *in* Φ *such that* T *proves* $\forall \vec{x} \exists! y\ \psi(\vec{x}, y)$*; we say that* ψ *defines a function in* T*.*

Notice that we want the function or relation to be given by the formula on $\mathbb{N}$, and in the case of the functions we want to be able to prove it is a function in some theory (which implies that it has the right properties on $\mathbb{N}$). We will actually not be so concerned with definability on $\mathbb{N}$, though it will sometimes clarify the discussion to begin in the context of $\mathbb{N}$. Once we define the appropriate formula, we will only care that we can prove certain properties in the theory at hand, no longer concerned with $\mathbb{N}$.

Lemma 2.1.6 *Let* T *be a theory.*

- *For a new relation symbol* $\mathtt{R}(\vec{x})$ *and a formula* $\psi(\vec{x})$*,* $\mathrm{T} + (\mathtt{R}(\vec{x}) \Leftrightarrow \psi(\vec{x}))$ *is a conservative extension of* T*.*
- *Suppose* $\psi(\vec{x}, y)$ *defines a function in* T*. Then for a new function symbol* $\mathtt{f}$*, the theory* $\mathrm{T} + (\mathtt{f}(\vec{x}) = y \Leftrightarrow \psi(\vec{x}, y))$ *is a conservative extension of* T*.*

Note that the above theorem does not state that these function and relation symbols can be used in the induction axioms. However sometimes this is the case, which is especially useful.

Definition 2.1.7 *(**Relativizing**) Let* $\mathtt{R}$ *be a new function or relation symbol which we "relativize" with respect to.*

- *By* $\Delta_0(\mathtt{R})$ *we mean the set of* Δ_0 *formulae in the language augmented by* $\mathtt{R}$.
- *By* $\mathrm{I}\Delta_0(\mathtt{R})$ *we mean the theory* $\mathrm{BASIC}^0 + \Delta_0(\mathtt{R})\text{-}\mathit{IND}$.

We present a number of results (here and in later sections), indicating when a relativized extension is in fact a conservative extension.

Theorem 2.1.8 *(Relativized Extensions)*

- *If* $\psi(\vec{x}, y)$ *is a* Δ_0 *formula defining a function in* $\mathrm{I}\Delta_0$*, then for a new function symbol* $\mathtt{f}$*,* $\mathrm{I}\Delta_0(\mathtt{f}) + (\mathtt{f}(\vec{x}) = y \Leftrightarrow \psi(\vec{x}, y))$ *is a conservative extension of* $\mathrm{I}\Delta_0$.
- *If* $\psi(\vec{x})$ *is a* Δ_0 *formula then for a new relation symbol* $\mathtt{R}$*,* $\mathrm{I}\Delta_0(\mathtt{R}) + (\mathtt{R}(\vec{x}) \Leftrightarrow \psi(\vec{x}))$ *is a conservative extension of* $\mathrm{I}\Delta_0$.

Thus when functions or relations have such definitions we will use them freely. For simple functions we will typically use them with little or no discussion, though we are really employing this theorem.

Now consider strengthening $\mathrm{I}\Delta_0$. The following axiom says that the exponentiation function is defined for all numbers.

Definition 2.1.9 *Let* EXP *be the axiom:* $\forall x, y\ \exists z\ \mathsf{exp}(x, y, z)$.

The axiom EXP is too strong for our use, except in a few cases where we show a principle is difficult due to its equivalence to EXP. Rather than adding EXP we will consider adding the weaker axiom Ω_1, alluded to earlier. Formally, Ω_1 is the statement $\forall x \exists z\ \mathsf{exp}(x, \log x, z)$, or more informally, "$x^{\log x}$ exists;" by $\log x$ we always mean $\lfloor \log_2 x \rfloor$ unless stated otherwise. We noted that an interesting extension of $\mathrm{I}\Delta_0$ is the theory $\mathrm{I}\Delta_0 + \Omega_1$. This theory is essentially equivalent to S_2, which we now go on to discuss in the next section.

2.2 The Theory S_2

The work of this section is all from Buss [9]. S_2 is the theory of bounded arithmetic which basically consists of the theory $\mathrm{I}\Delta_0$ extended by the faster growing function "#" (called **smash**). The

growth rate of this function is what gives the correspondence between S_2 and the polynomial time hierarchy (this will be discussed later). The language is $\{+, *, \#, |\cdot|, \lfloor \cdot/2 \rfloor, 0, 1\}$. The function $|x|$ is the length of the binary representation of x, or essentially $\log_2 x$ (for example: $|5| = 3$ since the binary representation of 5 is 101). The intended meaning of the binary function $\#$ is that $a\#b = 2^{|a|*|b|}$. Note that $n\#n \approx n^{\log n}$ (they are virtually equal), which is why S_2 is essentially the same as $I\Delta_0 + \Omega_1$.

In S_2 we can define bounded formulae as we did for $I\Delta_0$, though the language is different. We let Σ^b_∞ refer to the bounded formulae in the language of S_2. We can now go on to define S_2 by specifying some basic axioms and the allowed induction axioms.

Definition 2.2.1 *Let* BASIC^1 *be the usual set of 32 defining axioms from Buss [9] (it includes* BASIC^0*).*

Definition 2.2.2 *Let* S_2 *be the theory consisting of* $\mathrm{BASIC}^1 + \Sigma^b_\infty\text{-}IND$.

We will be interested in subtheories of S_2, which will be defined by further restricting the induction axioms to bounded formulae with only a limited amount of bounded quantifier alternation. When we count the alternation of bounded quantifiers, we will not count certain kinds of bounded quantifiers called **sharply bounded quantifiers**, by which we mean bounded quantifiers in which the bounding term is of the form $|t|$ where t is a term in the language (for example, "$\forall x < |a + b|$" is a sharply bounded quantifier). Given a number $i \geq 0$, Σ^b_i is roughly the collection of bounded formulae which begin with a bounded existential quantifier and then have i bounded quantifier alternations, *not counting* the sharply bounded quantifiers; notice that the "b" just indicates "bounded." Π^b_i is defined similarly, except that such formulae begin with a universal quantifier. For example, the following formula $\psi(p)$ is a Π^b_1 expression that p is prime:

$$p > 1 \wedge \forall x < p \ (x = 1 \ \vee \ \neg \exists y < p \ (y * x = p)).$$

Definition 2.2.3

1. $\Pi^b_0 = \Sigma^b_0$ *is the set of formulae all of whose quantifiers are sharply bounded.*
2. Σ^b_{i+1} *is defined inductively by:*
   - $\Pi^b_i \subseteq \Sigma^b_{i+1}$.
   - *If* ψ *is in* Σ^b_{i+1} *then so are* $\exists x \leq t \ \psi$ *and* $\forall x \leq |t| \ \psi$.
   - *If* ψ *and* ϕ *are in* Σ^b_{i+1} *then so are* $\psi \wedge \phi$ *and* $\psi \vee \phi$.
   - *If* ψ *is in* Σ^b_{i+1} *and* ϕ *is in* Π^b_{i+1} *then* $\neg\phi$ *and* $\phi \Rightarrow \psi$ *are in* Σ^b_{i+1}.

3. Π^b_{i+1} *is defined inductively by:*

- $\Sigma^b_i \subseteq \Pi^b_{i+1}$.
- *If* ψ *is in* Π^b_{i+1} *then so are* $\forall x \leq t\ \psi$ *and* $\exists x \leq |t|\ \psi$.
- *If* ψ *and* ϕ *are in* Π^b_{i+1} *then so are* $\psi \wedge \phi$ *and* $\psi \vee \phi$.
- *If* ψ *is in* Π^b_{i+1} *and* ϕ *is in* Σ^b_{i+1} *then* $\neg\phi$ *and* $\phi \Rightarrow \psi$ *are in* Π^b_{i+1}.

4. Σ^b_{i+1} *and* Π^b_{i+1} *are the smallest sets satisfying these properties.*

We can now define natural subtheories of S_2.

Definition 2.2.4 *For* $i \geq 0$*, let* T^i_2 *be the theory consisting of* $\mathrm{BASIC}^1 + \Sigma^{\mathrm{b}}_{\mathrm{i}}$*−IND.*

It turns out that subtheories defined with an alternative form of induction are interesting. We define an apparently weaker kind of induction axiom, **length induction** (provably equivalent to a form of induction called **polynomial induction**).

Definition 2.2.5 *Given a formula* ψ*, let* ψ*−LIND be:*

$$\psi(0) \wedge \forall x\ (\psi(x) \Rightarrow \psi(x+1)) \Rightarrow \forall x\ \psi(|x|)$$

We can now define more subtheories of S_2.

Definition 2.2.6 *Let* S^i_2 *be the theory consisting of* $\mathrm{BASIC}^1 + \Sigma^{\mathrm{b}}_{\mathrm{i}}$*−LIND.*

This gives us a hierarchy within S_2.

Theorem 2.2.7

- $\mathrm{S}^0_2 \subseteq \mathrm{T}^0_2 \subseteq \mathrm{S}^1_2 \subseteq \mathrm{T}^1_2 \subseteq \ldots \mathrm{S}_2$.
- $\mathrm{S}_2 = \bigcup_{i \in \mathbb{N}} \mathrm{T}^i_2 = \bigcup_{i \in \mathbb{N}} \mathrm{S}^i_2$.

It should be noted that it is a major open question as to whether or not the hierarchy of theories in S_2 is actually proper (it is only known that $\mathrm{S}^0_2 \neq \mathrm{T}^0_2$).

As before it will be useful to discuss relativized extensions of theories. Given a new symbol `f`, by $\Sigma^b_i(\mathtt{f})$ we mean the class of formula defined like Σ^b_i with `f` added to the language; we can give such relativized definitions for other formulae classes with similar kinds of definitions. So the relativized theory $\mathrm{S}^i_2(\mathtt{f})$ is $\mathrm{BASIC}^1 + \Sigma^{\mathrm{b}}_{\mathrm{i}}(\mathtt{f})$−LIND; other relativized extensions are defined similarly. We now discuss some useful relativized extensions which are conservative extensions.

Theorem 2.2.8 *If $\psi(\vec{x}, y)$ is a Σ^b_1 (or Π^b_1) formula defining a function in S^1_2, then for the theory T (either S^i_2 or T^i_2, for $i \geq 1$), and a new function symbol $\mathtt{f}$, $\mathrm{T}(\mathtt{f}) + (\mathtt{f}(\vec{x}) = y \Leftrightarrow \psi(\vec{x}, y))$ is a conservative extension of T.*

A similar fact will hold for relations of less complexity.

Definition 2.2.9 *Let T be some theory. A formula ψ is Δ^b_i in T if it is Σ^b_i and there is a Π^b_i formula ϕ such that T proves $\psi \Leftrightarrow \phi$.*

Theorem 2.2.10 *If $\psi(\vec{x})$ is Δ^b_1 in S^1_2, then for the theory T (either S^i_2 or T^i_2, for $i \geq 1$) and a new relation symbol $\mathtt{R}$, $\mathrm{T}(\mathtt{R}) + (\psi(\vec{x}) \Leftrightarrow \mathtt{R}(\vec{x}))$ is a conservative extension of T.*

Note that Parikh's theorem applies to S_2 and its subtheories, so the exponentiation function cannot be defined. We in fact have Parikh's theorem (theorem 2.1.4) with S^0_2 replacing $\mathrm{I}\Delta_0$.

Theorem 2.2.11 *(Parikh's Theorem for S_2) Suppose T is a first-order bounded theory containing S^0_2 and $\psi(\vec{x}, y)$ is a bounded formula such that T proves $\forall \vec{x} \exists y\ \psi(\vec{x}, y)$. Then there is a term t, not containing y, such that T proves $\forall \vec{x} \exists y < t\ \psi(\vec{x}, y)$.*

The theories contained in S_2 are all bounded theories, so Parikh's theorem applies to them.

Another central feature is the correspondence between S_2 and the polynomial time hierarchy. We now discuss some aspects of that connection, assuming some basic knowledge of computational complexity theory. In complexity theory, we define the complexity of a function or relation in terms of the "length" of the arguments. In some contexts the arguments are actually given as binary strings, so we simply take the length as the length of this binary string. We will be working with natural numbers, and so we will define our complexity in terms of the lengths of the numbers, that is, given a number x, we consider $|x|$, and given a list of input numbers $\vec{x} = \langle x_1, \ldots, x_k \rangle$, we consider $\langle |x_1|, \ldots, |x_k| \rangle$, where this latter list of lengths is abbreviated $|\vec{x}|$. First we recall the usual definition of the **polynomial time hierarchy** of predicates where our arguments are lists of natural numbers.

Definition 2.2.12 *(Polynomial Time Hierarchy)*

1. *Let Σ^p_1, the **NP relations**, be the set of relations $\mathbf{R}(\vec{x})$ such that there is a polynomial p and a non-deterministic Turing machine M, with the property that for any $\vec{x}$, M non-deterministically decides in time $p(|\vec{x}|)$ whether or not $\mathbf{R}(\vec{x})$ holds.*
2. *For $i > 1$, let Σ^p_i be the set of relations computable with an NP relation that has access to a Σ^p_{i-1} oracle.*

Theorem 2.2.13 *Let $i \geq 1$. A relation is in Σ_i^p iff it is Σ_i^b definable.*

Note that there is no issue of provability in this claim; the Σ_i^b formulae are simply viewed as relations on $\mathbb{N}$. We will in fact be more interested in a polynomial time hierarchy of functions which can be connected to provability.

Definition 2.2.14

1. *Let $\square_1^p$, the **polynomial time functions**, be the set of functions $\mathbf{f}(\vec{x})$ such that there is a polynomial p and a Turing machine M, with the property that for any $\vec{x}$, M computes the function value $\mathbf{f}(\vec{x})$ in time $p(|\vec{x}|)$.*
2. *For $i > 1$, let $\square_i^p$ be the set of functions in $\square_1^p$ that have access to an oracle from Σ_{i-1}^p.*

Now we can state Buss' well-known **witnessing theorem**, which builds on the work of Cook [11]. There are a number of theorems in bounded arithmetic of this sort, stating that if a particular theory can prove that something exists with a formula description of a certain complexity, then we can find a computational procedure of some complexity that actually finds such an object for us.

Theorem 2.2.15 *(Buss [9]) Let $i \geq 1$. A function is in $\square_i^p$ iff it can be Σ_i^b defined in S_2^i.*

Note that for the case of $i = 1$, we get that the polynomial time functions (i.e. $\square_1^p$) correspond exactly to the functions that are Σ_1^b definable in S_2^1. Recall theorem 2.2.8 which states that we can conservatively add to S_2^1 all its Σ_i^b defined functions. Thus we can conservatively add all the polynomial time functions to S_2^1. Such facts about S_2^1 are why it is taken to be one of the most significant theories of bounded arithmetic.

The aspect of the above theorem referred to as "witnessing" is the claim that a Σ_i^b defined function in S_2^i is in fact a $\square_i^p$ function. One of the significant applications of this is to show the independence of some theorem. Suppose for example that we want to show that S_2^1 does not prove the sentence $\forall x \exists y\ \psi(x, y)$, where ψ is of complexity Σ_1^b. If S_2^1 did prove it, then by the witnessing theorem, we would have a polynomial time function $\mathbf{f}(x)$ such that for all natural numbers n, $\psi(n, \mathbf{f}(n))$ holds (it is a small but useful technical point that we don't in fact need a unique y to apply this theorem). So if we can show that there is no such polynomial time algorithm, then we have the desired independence. Typically such independence results either assume a complexity class separation or apply to a relativized theory $\mathrm{S}_2^i(\mathtt{R})$. In the case of $\mathrm{S}_2^i(\mathtt{R})$, the witnessing theorem relativizes, meaning that the $\square_i^p$ function works for any oracle $\mathbf{R}$ substituted for $\mathtt{R}$; to show independence amounts to finding a troublesome oracle $\mathbf{R}$.

2.3 The Pigeonhole Principle

Now we discuss one of the most useful principles connected with bounded arithmetic, the **pigeonhole principle**. Informally, the pigeonhole principle (PHP_n^m) says that a function mapping m to n (recall that this means a mapping from the set $[m] = \{0, 1, \ldots, m-1\}$ to the set $[n] = \{0, 1, \ldots, n-1\}$) is not injective. A typical choice for m is $n+1$, yielding the usual pigeonhole principle. The function will be described using a new 2-place relation symbol $\mathtt{R}$, where the first argument is taken to be the function argument, and the second is the value of the function. Formally, we have two variants to consider.

Definition 2.3.1

- Functional Form:

  Let $\mathtt{R}$ *be a 2 place relation symbol. Then* $\mathit{fPHP}_n^m(\mathtt{R})$ *is the formula*

  $$\forall x < m \; \exists! d < n \; \mathtt{R}(x, d) \;\Rightarrow\; \exists x < y < m \; \exists d < n \; \mathtt{R}(x, d) \wedge \mathtt{R}(y, d).$$

- Relational (or multi-function) Form:

  Let $\mathit{rPHP}_n^m(\mathtt{R})$ *be the same except that the* $\exists!$ *quantifier is replaced by* $\exists$.

When we write PHP without a prefix of "r" or "f" we mean for our comments or claims to refer to both versions.

Now we consider provability of $\mathsf{PHP}_n^m(\mathtt{R})$ for various values of m, working in theories relativized with respect to the relation symbol $\mathtt{R}$. For the usual pigeonhole principle, with $m = n+1$, there is the following significant independence result.

Theorem 2.3.2 *[6]* $\mathrm{S}_2(\mathtt{R})$ *does not prove* $\mathit{PHP}_n^{n+1}(\mathtt{R})$.

This result extended the work of Ajtai [1], which showed the independence in the weaker theory of $\mathrm{I}\Delta_0(\mathtt{R})$. However a form of the pigeonhole principle, PHP_n^{2n}, called the **weak pigeonhole principle** is more easily proven. Paris, Wilkie, and Woods [39] first proved the weak pigeonhole principle in the context of bounded arithmetic. Their results were strengthened by Maciel, Pitassi and Woods [35] to obtain the following result.

Theorem 2.3.3 *For any* $\Sigma_1^b(\mathtt{R})$ *formula* $\psi(\mathtt{R})$, $\mathrm{T}_2^2(\mathtt{R})$ *proves* $\mathit{rPHP}_n^{2n}(\psi(\mathtt{R}))$.

We emphasize that the relational form is provable, which implies that the weaker functional form is provable in the same theory. A common application of this theorem is to simply take $\psi(\mathtt{R})$ to

be $\mathtt{R}$. This result is sharp in terms of the standard hierarchy of theories in S_2. Since Krajíček [31] showed that $S^2_2(\mathtt{R})$ does not prove even the weaker function form $\mathsf{fPHP}^{2n}_n(\mathtt{R})$ (see also [33], p. 216). Note that in the weaker theory $I\Delta_0(\mathtt{R})$ even the provability of the weak pigeonhole principle is an open question.

We also note that the various forms of the PHP^m_n for different values of m are essentially equivalent in a strong enough theory of bounded arithmetic. The following theorem essentially comes from [39]. Provable connections of this sort are explicitly mentioned in [34, theorem 6.1] and [46, lemma 2.1]. We state the particular form we will need later.

Theorem 2.3.4 *Let $i \geq 1$ and $\phi(\mathtt{R})$ be a formula of complexity $\Sigma^b_i(\mathtt{R})$; let $b(n)$ be a term, and $\epsilon > 0$ be a standard rational. Then there is a formula $\psi(\mathtt{R})$ of complexity $\Sigma^b_i(\mathtt{R})$ such that $S^i_2(\mathtt{R})$ proves $\mathit{PHP}^{b(n)}_n(\psi(\mathtt{R})) \Rightarrow \mathit{PHP}^{(1+\epsilon)n}_n(\phi(\mathtt{R}))$.*

Given the above theorem we see that as long as m is a little bit larger than $n+1$, the principles all collapse in the appropriately strong theory. Thus we will often only be interested in referring to any one of these principles. However, to be specific we use the common notation of $\mathsf{rWPHP}(\mathtt{R})$ to refer to $\mathsf{rPHP}^{2n}_n(\mathtt{R})$, and $\mathsf{fWPHP}(\mathtt{R})$ to refer to $\mathsf{fPHP}^{2n}_n(\mathtt{R})$; by $\mathsf{WPHP}(\mathtt{R})$ we mean either one. In both cases, we freely use the above theorem to strengthen our applications of WPHP.

We will have cause to use these principles in a number of contexts. One modification is to substitute a formula ψ (with 2 free variables) for the relation symbol $\mathtt{R}$, to obtain $\mathsf{PHP}^m_n(\psi)$. For a set of formula Φ, we use $\mathsf{PHP}^m_n(\Phi)$, to denote the set of formulae $\{\mathsf{PHP}^m_n(\psi) \mid \psi \in \Phi\}$; such formulae will typically be added as an axiom schema to a theory. Notice that the provability of the schema version of the weak pigeonhole principle (for bounded formulae) follows from the provability of $\mathsf{WPHP}(\mathtt{R})$ for a relation symbol $\mathtt{R}$. While S_2 cannot prove $\mathsf{PHP}^{n+1}_n(\mathtt{R})$, the provability of $\mathsf{PHP}^{n+1}_n(\Sigma^b_\infty)$ is unknown.

We will also use the pigeonhole principle in the second-order context (discussed in chapter 4) with a set variable in place of the relation symbol $\mathtt{R}$.

2.4 Coding Issues

Theories of arithmetic have natural numbers as their intended objects. However, natural numbers can be used to code all kinds of objects, including one of the most basic objects, sequences of numbers. Using such sequences, we will be able to talk about things like trees and colorings of graphs. We will discuss such codings in general before pointing out the differences between working in $I\Delta_0$ and S^1_2 (unless we say otherwise we are working simultaneously in both theories). We will not give that much detail since this has been worked and re-worked (see [9] for S^1_2 and [27] for $I\Delta_0$).

Sequences will be coded by viewing numbers as their binary expansion. Certain numbers will

have binary expansions of the correct form, and so given such a number s, it will satisfy a Δ_0 formula we call $\mathsf{seq}(s)$. We can then do a number of natural operations on sequences. If $\mathsf{seq}(s)$ holds and n is a number, then there is a number t such that $\mathsf{seq}(t)$ holds, and t encodes the sequence s with the number n appended to it. We can determine the length of a sequence, and given a number s coding a sequence, we can find the i^{th} entry in s, denoted s_i. The heuristic is that we can prove any basic facts about sequences as long as they do not assert the existence of any sequences or numbers that are "too big."

We now consider the important issue of how big our objects can grow. This is in fact the essential difference between sequences in $\mathrm{I}\Delta_0$ and S^1_2. Basically, we can work with any sequences whose encoding is a number we know to exist. The coding of a sequence of length y, containing elements $< x$ can be naturally coded by a number of size about x^y. Thus to talk about such sequences we need to know that x^y exists; more precisely, we need to know that $\exists z\ \mathsf{exp}(x, y, z)$ holds, which we sometimes abbreviate $\mathsf{exists}(x^y)$. In $\mathrm{I}\Delta_0$, if we know that x exists then we can form sequences with elements $< x$ of length k, where k is standard, since we know x^k exists. However suppose we want a length $\log x$ sequence. In S^1_2 this is fine, because $x^{\log x}$ exists, but this is a problem for $\mathrm{I}\Delta_0$. Length x sequences are a problem for both theories, because without the totality of the exponentiation function we do not know that x^x exists. Sequences in S^1_2 are relatively robust. However we will have to be careful in $\mathrm{I}\Delta_0$, checking that the necessary operations do not require the existence of numbers beyond our grasp.

A useful notion for S^1_2 is:

Definition 2.4.1 *Let $\mathsf{small}(y)$ abbreviate $\exists m\ y < |m|$.*

Note that S^1_2 proves $\exists z\ \exp(x, y, z) \Leftrightarrow \mathsf{small}(y)$. If a number y is small we are free to use sequences of this length in S^1_2.

To improve readability we will often simply refer to exponential size terms, which means that we are implicitly assuming their existence. So for example if we assert that $\phi(x^y)$, we mean $\exists z\ \exp(x, y, z) \wedge \phi(z)$. Using the existence of n^k we can code a number of objects. One issue is with referring to $\binom{n}{k}$. We can compute $n!$ using a length n sequence s in which $s_0 = 1$ and $s_i = i * s_{i-1}$, for $i > 0$; so $n!$ is just the value of s_n. This length n sequence contains large numbers bounded by n^n, so s is coded by a number of size about $(n^n)^n = n^{n^2} = 2^{n^2 \log n} = 2^n \# 2^n \# n$. Thus in S^1_2, as long as n is small (so 2^n exists) we can deal with $\binom{n}{k}$ in this way. However in $\mathrm{I}\Delta_0$, knowing that 2^n exists does not guarantee the existence of 2^{n^2}. In $\mathrm{I}\Delta_0$ this approach would require the additional assumption that 2^{n^2} exists. We can avoid this by using a result of D'Aquino [14]. Let $\mathbf{choose}(n, k, y)$ be the relation on $\mathbb{N}$ that holds if $y = \binom{n}{k}$. D'Aquino gives a Δ_0 definition of this relation and furthermore shows that $\mathrm{I}\Delta_0$ can prove a number of its basic properties. When we say that $\binom{n}{k}$ exists in the context of $\mathrm{I}\Delta_0$ (or its second order extensions), we mean that $\exists y\ \mathsf{choose}(n, k, y)$. We point out some important facts about this relation.

Lemma 2.4.2 *[14, lemma 20]* $\mathrm{I}\Delta_0$ *proves* $k\binom{n}{k} = \binom{n}{k-1}(n-k+1)$.

We prove the following three lemmas by induction and the use of the above lemma; we give the proof for just the first of the following three lemmas.

Lemma 2.4.3 $\mathrm{I}\Delta_0$ *proves that if* $\binom{n}{k}$ *exists then* $(\frac{n-k}{k})^k$ *exists and* $(\frac{n-k}{k})^k \leq \binom{n}{k}$.

Proof

Proceed by induction on k on the formula:

$$\mathsf{choose}(n,k,y) \Rightarrow \exists x \leq y\ \mathsf{exp}(\frac{n-k}{k}, k, x).$$

Consider the inductive step:

$$\begin{aligned}
\binom{n}{k} &= \frac{n-k+1}{k}\binom{n}{k-1} && \text{by lemma 2.4.2} \\
&\geq \frac{n-k+1}{k}\left(\frac{n-k+1}{k-1}\right)^{k-1} && \text{by inductive hypothesis} \\
&\geq \left(\frac{n-k}{k}\right)^k
\end{aligned}$$

□

Lemma 2.4.4 $\mathrm{I}\Delta_0$ *proves that if* n^k *exists then* $\binom{n}{k}$ *exists and* $\binom{n}{k} \leq n^k$.

We know that if x^y exists, then so do smaller powers. We want something similar to hold for the choose operator.

Lemma 2.4.5 $\mathrm{I}\Delta_0$ *proves that if* $\binom{n}{k}$ *exists and* $j < k \leq n/2$, *then* $\binom{n}{j}$ *exists.*

For the last lemma, notice that $\binom{n}{k}$ is increasing until $k = n/2$, so $\binom{n}{k}$ is large enough to bound the appropriate terms in the inductive proof; if $k > n/2$, $\binom{n}{k}$ could become too small. Since we often need the existence of the $\binom{n}{j}$ for j smaller than some k, we will sometimes restrict claims to the case of $k \leq n/2$; of course this claim also holds if we add a standard number to k, so we just need k near $n/2$ to apply this lemma.

We will sometimes want to encode a sequence in a very compact way. Suppose we have a sequence $\langle a_{k-1}, \ldots, a_0 \rangle$, where $a_i < n$, so it is coded by a number of size around n^k; in fact the

number is a little bigger in order to code delimiters between elements in the sequence. As a more compact way, we could associate this sequence with its **base n representation**, which is just the number $\sum_{i<k} a_i n^i$. All the n^k length k sequence with elements $< n$ are associated with a unique number $< n^k$; note that moving to the left, the entries become more significant, which will always be our convention. We can not get a general notion of sequences this way, since we do not know where one element ends and the next begins, but for particular applications we will sometimes wish to associate sequences with their base n representation. Note that pairing is a special case of this (for $k = 2$).

Chapter 3

Probabilistic Methods in Bounded Arithmetic

A number of techniques in combinatorics go by the name "probabilistic methods." A common feature is that they use probability theory to prove theorems of combinatorics, where these theorems (mostly) do not explicitly mention anything about probability. We consider formalizing these theorems in the first-order context of S_2. To formalize such techniques in bounded arithmetic we need to avoid explicit mention of probability, since this requires us to refer to the exponentially large probability spaces. It is often pointed out that such methods can just be seen as counting arguments which do not make reference to any probability theory. Though this kind of conversion will be a first step in our formalization, we are still stuck with exponentially many sets to count. Such difficulties will be discussed more explicitly for a concrete case that we formalize in section 3.1. The basic idea behind formalizing the probabilistic methods in bounded arithmetic is to use the weak pigeonhole principle to simulate the probabilistic counting argument, sidestepping the difficulty.

In section 3.1 we consider the "ordinary" probabilistic method, applied to three examples: A Ramsey lower bound, dominating sets in tournaments, and 2-coloring hypergraphs. By the ordinary probabilistic method we mean that an object is shown to exist by showing it exists with non-zero probability.

In section 3.2 we consider the method called "linearity of expectations," applied to two theorems. In this method one upper or lower bounds the number of objects of some kind by finding the expectation of some random variable. We can again use the weak pigeonhole principle to formalize the counting argument. We then discuss how the algorithmic method of "conditional probabilities" can be used to strengthen the theorems.

In section 3.3 we consider the Local Lemma, essentially an extension of the ordinary probabilistic method to the case in which the events in question have rare dependencies. We apply this to another

example of 2-coloring a hypergraph, one which cannot be solved by the ordinary probabilistic method. We make partial progress in this direction, offering some speculation on how to fully solve the problem.

We will reference the originators for their work, though the results are conveniently collected together in the book "The Probabilistic Method" [4].

Before moving into the results we should point out a relevant theorem. The various formalizations to follow will be carried out in $S_2^1 + \mathsf{WPHP}(\Sigma_1^b)$. The following is an unpublished result of Wilkie presented by Krajíček [33] (slightly modified to match our context).

Theorem 3.0.6 *(Wilkie) Let $\psi(x, y)$ be a Σ_1^b formula and suppose $S_2^1 + \mathsf{rWPHP}(\Sigma_1^b)$ proves $\forall x \exists y\, \psi(x, y)$. Then there is a bounded error polynomial-time probabilistic function $\mathbf{f}$ such that $\psi(x, \mathbf{f}(x))$ holds.*

To say that a function $\mathbf{f}$ is **bounded error polynomial-time probabilistic** (for short just "probabilistic function") we mean that in polynomial time it gives an output which is correct with probability $\geq 3/4$. We will discuss this theorem in relation to some of the concrete cases to follow.

3.1 Ordinary Probabilistic Method

The ordinary probabilistic method shows the existence of an object as follows. We consider some sample space Ω with a probability distribution. A number of events $A_1, \ldots, A_m \subset \Omega$ are the "bad" events, meaning that as long as an object from the sample space is not contained in any A_i, the object has the desired properties, or is "good." We then show that $\mathsf{prob}(A_1) + \ldots + \mathsf{prob}(A_m) < 1$ which means that the good object is in the sample space; thus we have an existence argument. This can be restated as a counting argument. Let N be the number of elements in the sample space Ω. Then showing $\|A_1\| + \ldots + \|A_m\| < N$, allows us to arrive at the same conclusion (here, by $\|A_i\|$ we mean the size of the set A_i).

In bounded arithmetic, neither argument will formalize directly when N is not small, since then the subsets $A_i \subset \Omega$ are too large to code as numbers. Our way around this is to avoid explicitly talking about these subsets. Instead we define a function (with a bounded formula) from the number N to $(1/2)N$; if the good object did not exist then the function would be injective, violating the weak pigeonhole principle. This function will mimic the counting argument by essentially grouping together the A_i without explicitly referring to them.

Thus a key issue in all of these proofs (the ordinary probabilistic method and the other probabilistic methods of the later sections) will be the fact that we can give a bounded definition of this function that mimics the counting argument. For this approach we need to be able to find a relatively constructive definition (i.e. definition by bounded formula), even though we are formalizing non-constructive probabilistic arguments. Thus to formalize a non-constructive existence

argument can be seen as showing that it has some degree of constructivity.

3.1.1 A Ramsey Lower Bound

Our first application of the ordinary probabilistic method is proving a Ramsey lower bound. We begin with some discussion of Ramsey theory; this will be used here and throughout the thesis when dealing with Ramsey theory.

For a number n, let K_n be the **complete graph on n vertices**, that is, a graph with n vertices and a single edge between any two distinct vertices. We use the conventional arrow notation $n \to (k)_r$ to mean that if each of the edges of K_n is assigned one of r colors then there is a size k subset X of the vertices, such that all of the edges with vertices in X are assigned the same color (X is called **monochromatic**). The assignment of colors is called an **r-coloring** if we are allowed to use r colors (we always assume $r \geq 2$). The smallest such n that makes the arrow relation hold is denoted $R_r(k)$.

Now we define the Ramsey principle. It uses a 3 argument relation symbol $\mathtt{H}$ to refer to the colors of edges; the first two arguments are vertices and the third argument is the color of the edge connecting them. The principle uses $[n]$ as its vertices and $[r]$ as its colors, saying that if $\mathtt{H}$ is a valid $r-$coloring, then there is a monochromatic set of the correct size. We use the capital letter X to denote a number that codes a set. It is coded as a sequence of non-repeating numbers. We use "size" as a more suggestive name to refer to the length of a sequence which encodes a set; we sometimes use $\|X\|$ to refer to the size of X when it is clear that we do not mean to be applying the "binary length" operator twice.

Definition 3.1.1 *Let* $\mathtt{H}$ *be a relation symbol with 3 arguments.*

- *Let* $\mathsf{coloring}(\mathtt{H}, n, r)$ *be:* $\forall x < y < n \; \exists! d < r \; \mathtt{H}(x, y, d)$.
- *Let* $\mathsf{monochromatic}(\mathtt{H}, X, n, r)$ *be:* $\exists d < r \; \forall u, v \in X \; \mathtt{H}(u, v, d)$.
- *Let* $\mathsf{Ramsey}(\mathtt{H}, n \to (k)_r)$ *be:*
  $\mathsf{coloring}(\mathtt{H}, n, r) \Rightarrow \exists X \subseteq [n] \; (\mathsf{size}(X) = k \wedge \mathsf{monochromatic}(\mathtt{H}, X, n, r))$.

In one of the first applications of the probabilistic method Erdős [21] showed that for $n = 2^{k/2}$, there exists a 2-coloring of K_n with no size k monochromatic set, or in other words $R_2(k) > 2^{k/2}$. To formalize this we will want to be able to refer to the existence of a coloring, so we will use numbers to code colorings. A $2-$coloring on n vertices will be coded by a number $G < \mathsf{pow}(2, \binom{n}{2})$, where $\mathsf{pow}(x, y)$ is just x^y. The number G can be interpreted as a binary string of length $\binom{n}{2}$, so each of the $\binom{n}{2}$ edges is colored "0" or "1" accordingly (often more intuitively referred to as "blue" or "red"); note that the number of vertices n must be small, which will be implicit throughout our

discussion. The above Ramsey principle is formalized with a relation symbol, but we can naturally substitute our number encoding for the relation symbol (later we use the Ramsey principle in its form with a relation symbol). Finally we formalize the claim by proving the following.

Theorem 3.1.2 $S^1_2 + \mathit{rWPHP}(\Sigma^b_1)$ *proves* $\exists G < \mathit{pow}(2, \binom{2^{k/2}}{2})\ \neg \mathit{Ramsey}(G, 2^{k/2} \to (k)_2)$.

Note that as a corollary T^2_2 proves the statement in the above theorem, as an immediate consequence of theorem 2.3.3. We will not mention this for future proofs that go through in $S^1_2 + \mathsf{rWPHP}(\Sigma^b_1)$; these comments also apply to $S^1_2 + \mathsf{fWPHP}(\Sigma^b_1)$ since the theory $S^1_2 + \mathsf{rWPHP}(\Sigma^b_1)$ is at least as strong.

To prove the theorem first consider the informal probabilistic argument restated as a counting argument. Call a coloring "bad" if it has a size k monochromatic set, and "good" otherwise. So our goal is to show that a good coloring exists. There are $\binom{n}{k}$ size k subsets of vertices and for each such subset there are $\mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1)$ colorings that make it monochromatic. Thus the number of bad colorings is bounded by $\binom{n}{k}\mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1) = \binom{n}{k}\mathsf{pow}(2, 1 - \binom{k}{2})\mathsf{pow}(2, \binom{n}{2})$. Since $n = 2^{k/2}$, a calculation shows that $\binom{n}{k}\mathsf{pow}(2, 1 - \binom{k}{2}) < 1$, so the number of bad colorings is less than $\mathsf{pow}(2, \binom{n}{2})$, which is the total number of colorings. So there must be a good coloring.

The lack of exponentiation in S_2 precludes formalizing this argument directly since the above proof involves counting large sets of colorings. To formalize the theorem we reformulate the proof, using the structure of the counting argument to define a multi-function on the set of all colorings, simulating the argument using rWPHP.

Suppose for sake of contradiction that the theorem does not hold. So for some small k and n, where we let $n = 2^{k/2}$, every coloring $G < \mathsf{pow}(2, \binom{n}{2})$ is bad. We now define a Σ^b_1 multi-function, F, from the set of all colorings (i.e. numbers $< \mathsf{pow}(2, \binom{n}{2})$) to a set which counts all the bad colorings (numbers bounded by $(1/2)\mathsf{pow}(2, \binom{n}{2})$). F will be an injective multi-function and so violate $\mathsf{rWPHP}(\Sigma^b_1)$.

First we sketch the definition of F. Given a coloring $G < \mathsf{pow}(2, \binom{n}{2})$, F will find a size k monochromatic set X in G, which exists by our assumption. The function **setNumber** (to be defined) will map the set X to s, $0 \leq s < \binom{n}{k}$; each set is mapped to a different number. Many colorings have X as its monochromatic set, so to uniquely identify G, we indicate which of the 2 colors X has, and for the remaining $\binom{n}{2} - \binom{k}{2}$ edges, we choose the appropriate coloring of the edges. This can be seen as a binary string of length $\binom{n}{2} - \binom{k}{2} + 1$, so we arrive at a number, say v, where $v < \mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1)$. We can obtain this v by a Σ^b_1 definable (in S^1_2) function rest, so $\mathsf{rest}(G, X) = v$. G is then mapped to $s * \mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1) + v$.

Now we define the function **setNumber**, a sort of lexicographic ordering which assigns set $\{0, 1, \ldots, k-1\}$ to 0, $\{n-k, \ldots, n-1\}$ to $\binom{n}{k} - 1$, and other sets to intermediate numbers in an injective manner.

Definition 3.1.3 *For $X \subseteq [n]$, let* **setNumber**$(X; n) = f_0(X; n)$*, where*

- $f_a(\{\}; n) = 0$ *for* $0 \leq a \leq n$
- $$f_a(X; n) = \begin{cases} f_{a+1}(X - \{a\}; n) & if\ a \in X, \\ \binom{n-a-1}{size(X)-1} + f_{a+1}(X; n) & otherwise. \end{cases}$$

This recursively defined function can be given by a Σ^b_1 formula using sequences of small size n. We now prove some properties about it in S^1_2. When n is clear from context, we may just write $f_a(x)$ instead of $f_a(x; n)$.

Lemma 3.1.4 S^1_2 *proves* $f_a(X) < \binom{n-a}{size(X)}$*, for* $X \subseteq \{a, \ldots, n-1\}$.

Proof

We show this by induction on a from n down to 0. For the inductive step we assume the claim for $a+1$ and then need to show $f_a(X) < \binom{n-a}{size(X)}$, where $X \subseteq \{a, \ldots, n-1\}$.

If $a \in X$, then $f_a(X) = f_{a+1}(X - \{a\}) < \binom{n-a-1}{size(X)-1} \leq \binom{n-a}{size(X)}$.

Otherwise, $a \notin X$. For $X = \{\}$ we are done. Otherwise we carry out the following calculation:

$f_a(X) = \binom{n-a-1}{size(X)-1} + f_{a+1}(X) < \binom{n-a-1}{size(X)-1} + \binom{n-a-1}{size(X)} = \binom{n-a}{size(X)}$.

□

The lemma shows that the range of setNumber (i.e. f_0) really is $[\binom{n}{k}]$, for size k subsets of $[n]$. We can also see that it is injective on size k subsets of $[n]$. Consider two distinct sets $X, Y \subseteq [n]$, both of size k. Let b be the smallest element in one set, but not in the other; suppose $b \in X$, and $b \notin Y$. Since the recursive procedure will be the same up to b, we get that $f_0(X) = m + f_b(X')$ and $f_0(Y) = m + f_b(Y')$, for some number m; the sets X' and Y' come from X and Y, respectively, with the same elements from $\{0, \ldots, b-1\}$ removed by the recursive procedure (so in particular, the size of the primed sets are also the same). It now suffices to show that $f_b(X') < f_b(Y')$. We

have

$$\begin{aligned} f_b(X') &= f_{b+1}(X' - \{b\}) \\ &< \binom{n-b-1}{size(X')-1} \\ &\leq f_{b+1}(Y') + \binom{n-b-1}{size(X')-1} \\ &= f_b(Y'). \end{aligned}$$

We can now define the multi-function F, with the coloring G as input, and y as the output.

Definition 3.1.5 *Let $F(G, y)$ be the following Σ_1^b formula*

$$\begin{aligned} \exists X \subseteq [n]\ \exists s, v \quad & \mathit{size}(X) = k & \wedge \\ & \mathit{monochromatic}(G, X, n, 2) & \wedge \\ & \mathit{setNumber}(X) = s & \wedge \\ & \mathit{rest}(G, X) = v & \wedge \\ & y = s * \mathit{pow}(2, \tbinom{n}{2} - \tbinom{k}{2} + 1) + v \end{aligned}$$

By the assumption there exists such X, so F is a multi-function. F is injective, since setNumber and rest are injective and $v < \mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1)$. To show that the range of F is $(1/2)\mathsf{pow}(2, \binom{n}{2})$ we prove the following bound (essentially from [4]).

Claim 3.1.6 S_2^1 *proves $y \leq (1/2)\mathit{pow}(2, \binom{n}{2})$.*

Proof

From the definition, $s \leq \binom{n}{k} - 1$ and $v < \mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1)$, so

$$\begin{aligned} y &< (\binom{n}{k} - 1)\mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1) + \mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1) \\ &= \binom{n}{k}\mathsf{pow}(2, \binom{n}{2} - \binom{k}{2} + 1) \\ &= \mathsf{pow}(2, \binom{n}{2})\binom{n}{k}\mathsf{pow}(2, 1 - \binom{k}{2}). \end{aligned}$$

We are done when we show that $\binom{n}{k}\mathsf{pow}(2, 1 - \binom{k}{2}) < 1/2$.

$$\binom{n}{k}\mathsf{pow}(2,1-\binom{k}{2}) < \frac{n^k}{k!}\frac{\mathsf{pow}(2,1+k/2)}{\mathsf{pow}(2,k^2/2)} \leq \frac{\mathsf{pow}(2,1+k/2)}{k!}\frac{(\mathsf{pow}(2,k/2))^k}{\mathsf{pow}(2,k^2/2)},$$

substituting $2^{k/2}$ for n.

The last expression is bounded by $1/2$ for most k ($k \geq 4$).

□

That finishes the entire proof since we have now arrived at a contradiction, namely, a multi-function F violating the weak pigeonhole principle.

The technique used in this proof provides a recipe for formalizing such non-constructive counting arguments, as long as the counting argument is constructive enough to allow such a function to be defined. The complexity of the function definition affects what theory suffices. We recall theorem 3.0.6 which gives us a probabilistic witnessing function. We have shown that $\mathrm{S}^1_2 + \mathsf{rWPHP}(\Sigma^b_1)$ proves $\exists G < \mathsf{pow}(2, \binom{2^{k/2}}{2})$ $\neg\mathsf{Ramsey}(G, 2^{k/2} \rightarrow (k)_2)$, but we cannot quite apply the witnessing theorem since the formula proven is Σ^b_2. The only quantifiers that matter (i.e. the rest are sharply bounded) are the "$\exists G$" and "$\neg\exists X \subseteq [n]$" which yield the indicated complexity. If we could by some trick bring the complexity of the formula down to Σ^b_1, we would obtain a probabilistic algorithm (polynomial in the number of vertices) to find G. In fact we already have such an algorithm: *Randomly color each edge red or blue, with probability* $1/2$. Preceding calculations (i.e. claim 3.1.6) show that this produces the desired coloring with probability $(1-\epsilon) \geq 3/4$ for sufficiently large k.

3.1.2 Dominating Sets in Tournaments

Now we consider some theorems of Erdős [23] concerning tournaments.

Definition 3.1.7

1. *A **tournament** is a directed graph in which any two distinct vertices have exactly one directed edge between them. If there is an edge starting at u and pointing at v, we say that u **beats** v.*
2. *A **dominating set** in a tournament is a set D of vertices such that for any vertex w not in D, there is a vertex $u \in D$ that beats it.*

Theorem 3.1.8 *(Erdős [23]) A tournament with n vertices has a dominating set of size* $\log n$.

The proof proceeds by choosing a vertex that beats at least half of the other vertices (true by an averaging argument). The beat vertices are thrown out and the process continues for at most $\log n$ steps.

To formalize this we let $[n]$ be our set of vertices. The tournament is exhibited by a two-place relation symbol $\mathtt{R}$: For $x \neq y < n$, $\mathtt{R}(x, y)$ holds if and only if x beats y. A dominating set is referred to by a number D coding a subset of $[n]$.

Definition 3.1.9

- *Let* $\mathsf{tournament}(\mathtt{R}, n)$ *be:* $\forall x < y < n \ (\mathtt{R}(x, y) \Leftrightarrow \neg\mathtt{R}(y, x))$.
- *Let* $\mathsf{dominating}(\mathtt{R}, D, n)$ *be:* $\forall v < n \ (v \in D \vee \exists u \in D \ \mathtt{R}(u, v))$.

Krajíček posed the question of whether or not a formalization of Erdős' theorem can be proved in bounded arithmetic (possibly weakening the theorem by letting the size of D be larger than $\log n$).

Question 3.1.10 *(Krajíček) Does some theory of bounded arithmetic prove the "Tournament Principle":*

$$\mathsf{tournament}(\mathtt{R}, n) \Rightarrow \exists D \subseteq [n] \ (\mathsf{dominating}(\mathtt{R}, D, n) \wedge \mathsf{size}(D) = \log n)?$$

Erdős also considered the dual question of showing that there exists a tournament that does not have a dominating set of a certain size.

Theorem 3.1.11 *[23] If* $\binom{n}{k}(1 - 2^{-k})^{n-k} < 1$ *then there is a tournament on* n *vertices with no size* k *dominating set.*

By a calculation he obtains a corollary.

Corollary 3.1.12 *[23] For any positive number* ϵ*, there exists a positive number* K*, such that whenever* $k > K$*, and* $n > 2^k k^2 \log_e(2 + \epsilon)$*, there is a tournament on* n *vertices that does not have a dominating set of size* k*.*

Restated this says that for any $\epsilon > 0$ and sufficiently large n there is a tournament on n vertices with no size $(1 - \epsilon) \log n$ dominating set; theorem 3.1.8 says that there is a dominating set of size $\log n$.

We will formalize the last two claims. Formalizing theorem 3.1.11 follows section 3.1.1 closely, though corollary 3.1.12 does not. In order to assert the existence of a tournament, we will code a tournament on n vertices with a number $T < 2^{\binom{n}{2}}$, in a manner similar to the last section on Ramsey theory. We view T as a length $\binom{n}{2}$ binary string, with each bit indicating which direction the edge points. Given $u, v < n$, by $T(u, v)$ we mean that u beats v. We can naturally substitute the number T for the relation symbol $\mathtt{R}$ in our definitions.

We first prove a theorem similar to theorem 3.1.11, requiring a bound of $1/2$ in order to apply rWPHP. Keeping the bound of 1 would require the full pigeonhole principle. The proof is similar to that of the Ramsey lower bound, theorem 3.1.2, so we will skip many details, pointing out the features unique to this problem.

Theorem 3.1.13 $S_2^1 + rWPHP(\Sigma_1^b)$ *proves that if* $\binom{n}{k}(1-2^{-k})^{n-k} \leq 1/2$ *then*

$$\exists T < 2^{\binom{n}{2}} \ (tournament(T,n) \wedge \neg\exists D \subseteq [n] \ (dominating(T,D,n) \wedge size(D) = k)).$$

Proof

We assume for contradiction that all the tournaments have size k dominating sets, and then define a multifunction F from the the set of all tournaments (i.e. numbers $T < 2^{\binom{n}{2}}$) to those with size k dominating sets. The multifunction $F(T,y)$ will be Σ_1^b, defined as follows:

$$\begin{array}{lll} \exists X \subseteq [n] \ \exists s,v & \mathsf{size}(X) = k & \wedge \\ & \mathsf{dominating}(T,X,n) & \wedge \\ & \mathsf{setNumber}(X) = s & \wedge \\ & \mathsf{rest}(G,X) = v & \wedge \\ & y = s * (2^k-1)^{n-k} * \mathsf{pow}(2, \binom{k}{2} + \binom{n-k}{2}) + v & \end{array}$$

setNumber is defined as in section 3.1.1, but rest is defined differently. Once we have a particular dominating set X we count the possible edge arrangements by considering 3 groups of edges: those with exactly 0,1,or 2 vertices in X. To count those with exactly one vertex in X we consider each of the $n-k$ vertices $u_1, \ldots, u_{n-k}$, outside of X. For each u_i we cannot have it beating all of X, so there are $2^k - 1$ ways of drawing edges between u_i and X; this relationship can be represented by a number $a_i < 2^k - 1$. Since we have $n-k$ u_i's there are $(2^k-1)^{n-k}$ ways to draw these edges; we represent this relationship by the sequence $(a_1, \ldots, a_{n-k})$ which we encode by its base 2^k-1 representation, a number $a < (2^k-1)^{n-k}$. There are $\binom{k}{2}$ edges with two vertices in X, coded by a binary string of that length, so a number $b < \mathsf{pow}(2, \binom{k}{2})$. Similarly, those edges with no vertices in X are coded by a number $c < \mathsf{pow}(2, \binom{n-k}{2})$. So $\mathsf{rest}(T,X) = abc < (2^k-1)^{n-k} * \mathsf{pow}(2, \binom{k}{2} + \binom{n-k}{2})$.

To see that F is injective and the range is as desired, we calculate some bounds.

From our definitions we know that $v < (2^k-1)^{n-k}\mathsf{pow}(2, \binom{k}{2} + \binom{n-k}{2})$. This combined with the definition of the number y ensures that the function F really is injective. Now the key calculation is the following.

Claim 3.1.14 S^1_2 *proves that* $y \leq (1/2)2^{\binom{n}{2}}$.

Proof

By substituting the largest allowed numbers for s and v, we see that $y < \binom{n}{k}(2^k-1)^{n-k}\mathsf{pow}(2, \binom{k}{2} + \binom{n-k}{2})$.
Now for a calculation that follows Erdős closely.

$$\begin{aligned}\binom{n}{k}(2^k-1)^{n-k}\mathsf{pow}(2, \tbinom{k}{2} + \tbinom{n-k}{2}) &= \binom{n}{k}(1-2^{-k})^{n-k}2^{k(n-k)}\mathsf{pow}(2, \tbinom{k}{2} + \tbinom{n-k}{2}) \\ &= \binom{n}{k}(1-2^{-k})^{n-k}2^{\binom{n}{2}} \\ &\leq (1/2)2^{\binom{n}{2}}.\end{aligned}$$

Note that the second equality follows by a calculation which shows that

$$k(n-k) + \binom{k}{2} + \binom{n-k}{2} = \binom{n}{2}.$$

□

We now see that F violates rWPHP, finishing the proof.

□

We now prove a formalization of corollary 3.1.12, which showed $n = 2^k k^2 \log_e(2+\epsilon)$ vertices is enough to avoid a size k dominating set. We will use 2 instead of e, and for ease of readability, we fix the constant $\epsilon = 2$, though we could choose $\epsilon > 0$ to be any standard rational.

Corollary 3.1.15 $S^1_2 + \mathit{rWPHP}(\Sigma^b_1)$ *proves for* $k \geq 7$ *and* $n = k^2 2^{k+1}$,

$$\exists T < 2^{\binom{n}{2}}\ (\mathit{tournament}(T,n) \wedge \neg\exists D \subseteq [n]\ (\mathit{dominating}(T,D,n) \wedge \mathit{size}(D) = k)).$$

A bound used by Erdős in this proof is: $1 - 2^{-k} < e^{-2^{-k}}$. This follows easily from the fact that $1 + x \leq e^x$, which can be proved by use of derivatives. To avoid dealing with e, we replace it by 2. We also replace the proof using calculus with a proof using arithmetic and induction, since it is not clear how much calculus we can do in bounded arithmetic.

Lemma 3.1.16 S^1_2 *proves for all* $k \geq 0$, $1 - 2^{-k} \leq 2^{-2^{-k}}$.

Proof

We first show two facts about bounds on the roots of numbers, and then apply this to prove the lemma.

First, for $x > 1$, $\sqrt{x} \leq \frac{1+x}{2}$ (this is generally true, though in bounded arithmetic we take x to be a rational). This is true because

$$\begin{aligned} \sqrt{x} \leq \frac{1+x}{2} \quad &\text{iff} \quad x - 2\sqrt{x} + 1 \geq 0 \\ &\text{iff} \quad (\sqrt{x} - 1)^2 \geq 0; \end{aligned}$$

this last expression is clearly true.

Second, we will show by induction on k, that $2^{2^{-k}} \leq 1 + 2^{-k}$, for $k \geq 0$. Consider the inductive step.

$$\begin{aligned} 2^{2^{-(k+1)}} &= \sqrt{2^{2^{-k}}} \\ &\leq \tfrac{1+2^{2^{-k}}}{2} && \text{, by the first fact} \\ &\leq \tfrac{1+1+2^{-k}}{2} && \text{, by inductive hypothesis} \\ &= 1 + 2^{-(k+1)} \end{aligned}$$

Now to prove the lemma.

$$\begin{aligned} 2^{-2^{-k}} &= \tfrac{1}{2^{2^{-k}}} \\ &\geq \tfrac{1}{1+2^{-k}} && \text{, by the second fact} \\ &\geq 1 - 2^{-k}, \end{aligned}$$

where that last inequality holds because

$$\frac{1}{1+2^{-k}} \geq 1 - 2^{-k} \text{ iff } 1 \geq (1+2^{-k})(1-2^{-k}) = 1 - 2^{-2k},$$

and the latter expression ($1 \geq 1 - 2^{-2k}$) is clearly true.

□

Now we prove the corollary which essentially follows Erdős' proof. We include it here for completeness and to show that the calculation works in S^1_2.

Proof

By theorem 3.1.13 it suffices to show that $\binom{n}{k}(1 - 2^{-k})^{n-k} \leq 1/2$ under the given constraints. We have:

- $\binom{n}{k} < \frac{n^k}{k!}$, immediately from the definition of the choose operator, and

- $(1-2^{-k}) \leq 2^{-2^{-k}}$, by lemma 3.1.16

Thus,

$$\binom{n}{k}(1-2^{-k})^{n-k} < \frac{n^k}{k!}2^{(-2^{-k})(n-k)} = \frac{2^{k2^{-k}}}{k!}n^k 2^{-n2^{-k}}.$$

Consider the last expression. It's first part $\frac{2^{k2^{-k}}}{k!} = \frac{(2^{2^{-k}})^k}{k!} \leq \frac{2^k}{k!} \leq 1/2$, for $k \geq 5$; Erdős ignores this term since it is < 1. Now we just need the latter part of the expression $n^k 2^{-n2^{-k}} \leq 1$.

$2 \geq 1 + (1/k)\log 2k^2$, for $k \geq 7$, which implies $2k \geq \log k^2 2^{k+1}$. Thus

$$\frac{1}{k2^k} = \frac{2k}{k^2 2^{k+1}} \geq \frac{\log k^2 2^{k+1}}{k^2 2^{k+1}} = \frac{\log n}{n}.$$

From $\frac{1}{k2^k} \geq \frac{\log n}{n}$ we obtain $n^k 2^{-n2^{-k}} \leq 1$.

□

3.1.3 2-coloring Hypergraphs

We consider a coloring problem for hypergraphs. Though it is similar to the previous two proofs, we will discuss an extension of it in section 3.3.

Definition 3.1.17

- *A **hypergraph** is a pair (V,E), where V is some finite set (called the **vertices** or the **ground set**) and E is a collection of subsets of V (these subsets are called **hyperedges**).*
- *A hypergraph is called **k-uniform** if all of its hyperedges are sets of size k.*
- *A **2-coloring** is a coloring of the ground set by 2 colors. A hypergraph is **2-colorable** if there is a 2-coloring, such that every hyperedge contains a vertex of both colors.*

Theorem 3.1.18 *(Erdős [22]) A k-uniform hypergraph with $< 2^{k-1}$ hyperedges is 2-colorable.*

The probabilistic argument can be seen as follows. We want to find a *good* coloring, where good means that every hyperedge contains vertices of both colors. For each hyperedge i, let A_i be the event consisting of all the *bad* colorings in which hyperedge i is monochromatic. The probability of some bad event occurring is < 1, therefore, there is a good coloring.

In considering how we are going to formalize this statement, notice that to say a hypergraph is 2-colorable requires asserting that a 2−coloring of the ground set exists. If the ground set is $[n]$, then we code a 2-coloring by a number $C < 2^n$ in the natural way, so n will be small; for $x < n$, by $C(x)$ we mean the color of vertex x. We also need to refer to the hypergraph, a collection of subsets of $[n]$. Coding the hypergraph as a number would require 2^n to be small, but this can be avoided by using a binary relation symbol $\mathtt{H}$ to refer to the hypergraph. $\mathtt{H}(i,x)$ will say that element $x < n$ is in the i^{th} hyperedge. Since the i^{th} hyperedge is $\{x < n \mid \mathtt{H}(i,x)\} \subseteq [n]$, and n is small, S^1_2 can code it as a number which we call "$\mathtt{H}_i$." We use "size" (again) to refer to the length of this sequence. To say a number C coding a coloring makes $\mathtt{H}_i$ monochromatic can be stated by $\mathsf{monochromaticHyper}(\mathtt{H}, C, i)$:

$$\forall x, y < n \ (\mathtt{H}(i,x) \wedge \mathtt{H}(i,y) \Rightarrow \ C(x) = C(y)).$$

We now formalize theorem 3.1.18 by the following weaker claim, which allows 2^{k-2} edges, rather than 2^{k-1}, in order to apply rWPHP. The stronger claim would use the full pigeonhole principle.

Theorem 3.1.19 $\mathrm{S}^1_2(\mathtt{H}) + \mathit{rWPHP}(\Sigma^b_1(\mathtt{H}))$ *proves*

$\forall i < 2^{k-2} \ (\mathit{size}(\mathtt{H}_i) = k \wedge \forall x(\mathtt{H}(i,x) \Rightarrow x < n)) \Rightarrow \exists C < 2^n \forall i < 2^{k-2} \ \neg \mathit{monochromaticHyper}(\mathtt{H}, C, i)$.

Proof

Assume for sake of contradiction that all the colorings are bad, each one making some hyperedge monochromatic. We define a function F from 2^n (the set of all colorings) to $(1/2)2^n$, which due to our assumption will be an injective multi-function. Given $C < 2^n$, $\exists i < 2^{k-2}$ $\mathsf{monochromaticHyper}(\mathtt{H}, C, i)$. That gives us a bounded existential quantifier in our definition of F, so the definition becomes $\Sigma^b_1(\mathtt{H})$ (notice monochromaticHyper uses sharply bounded quantifiers so we do not count such quantifiers).

Once we have found such an i, there are 2 ways to color the vertices of hyperedge $\mathtt{H}_i$ (all one color or all the other) and 2^{n-k} ways to color the remaining vertices; let $\mathsf{rest}(C)$ pick the appropriate one for C, so $\mathsf{rest}(C) < 2^{n-k+1}$. F maps C to:

$$\begin{aligned} i * 2^{n-k+1} + \mathsf{rest}(C) &< (2^{k-2} - 1)(2^{n-k+1}) + 2^{n-k+1} \\ &= 2^{k-2}2^{n-k+1} \\ &= (1/2)2^n. \end{aligned}$$

We have arrived at a contradiction because this violates rWPHP.

□

We have one concluding remark on the ordinary probabilistic method. The proofs all worked in S_2^1+rWPHP(Σ_1^b) by defining a Σ_1^b multi-function. They could also be carried out in S_2^1+fWPHP(Σ_2^b), since we can make a multi-function R into a function F by choosing defining $F(x)$ to be the smallest y such that $R(x, y)$. This leads to the following question.

Question 3.1.20 *Can the ordinary probabilistic method be formalized in a theory weaker than* S_2^1 + *rWPHP*(Σ_1^b)*? For example, does* S_2^1 + *fWPHP*(Σ_1^b) *suffice?*

We will pursue this question further in the conclusion.

3.2 Linearity of Expectations

We will consider another type of probabilistic method, called "Linearity of Expectations," given this name because a key step for this approach is that the expectation of a sum is equal to the sum of the expectations. As in the ordinary probabilistic method, we can use the weak pigeonhole principle to simulate the counting argument, although the use will follow a different pattern. We will then show how the weak pigeonhole principle can be eliminated by formalizing the "method of conditional probabilities," an algorithm which finds an object, rather than just showing existence. We will apply these methods to formalize two theorems.

3.2.1 Using the Weak Pigeonhole Principle

Recall that K_n is the complete graph on n vertices. There are a total of $\binom{n}{4}$ K_4 subgraphs (i.e. complete subgraphs on 4 vertices). The following theorem shows there is a coloring of the edges in which at most a fraction of these K_4 subgraphs are monochromatic.

Theorem 3.2.1 *[4] There is a 2-coloring of the edges of* K_n *such that the number of monochromatic* K_4 *subgraphs is at most* $\binom{n}{4}2^{-5}$.

For the probabilistic proof let X be the random variable giving the number of monochromatic K_4 subgraphs for a random 2-coloring of the edges, over the uniform probability space in which each coloring is equally likely. The expectation $E(X) = E(X_0 + \ldots + X_{r-1})$, where $r = \binom{n}{4}$ and X_i is the indicator random variable which is 1 if the i^{th} K_4 subgraph is monochromatic, and 0 otherwise.

By linearity of expectations we can calculate as follows:

$$\begin{aligned} E(X) &= E(X_0 + \ldots + X_{r-1}) \\ &= E(X_0) + \ldots + E(X_{r-1}) \\ &= 2^{-5} + \ldots + 2^{-5} \\ &= \binom{n}{4} 2^{-5} \end{aligned}$$

Thus there must in fact be a coloring with at most $\binom{n}{4}2^{-5}$ monochromatic K_4.

We formalize the statement of the theorem by letting a number $< 2^{\binom{n}{2}}$ represent an edge coloring as before. Thus n will be small, so we can code a sequence that refers to a set of $\binom{n}{4}$ K_4 subgraphs.

Theorem 3.2.2 $\mathrm{S}^1_2 + \mathit{fWPHP}(\Sigma^b_1)$ *proves*

$$\exists C < 2^{\binom{n}{2}} \forall S \ (S \textit{ is a set of monochromatic } K_4 \textit{ in } C) \Rightarrow \mathsf{size}(S) < \binom{n}{4} 2^{-5}.$$

As a first step to the formalization we restate the probabilistic argument as a counting argument, not yet concerned with bounded arithmetic.

> Consider a $2^{\binom{n}{2}} \times \binom{n}{4}$ matrix with the columns labeled by the all the K_4 subgraphs and the rows labeled by the 2-colorings of the edges. Put a 1 in entry (i, j) if the K_4 subgraph of column j is monochromatic for the coloring of row i, and a 0 otherwise. For each K_4, $2^{\binom{n}{2}}2^{-5}$ of the colorings make it monochromatic, so we have this many 1's per a column. Thus the total number of 1's in the matrix is $\binom{n}{4}2^{\binom{n}{2}}2^{-5}$. Now, assume for sake of contradiction that there is no coloring with $\binom{n}{4}2^{-5}$ or fewer monochromatic K_4; thus every row has at least $(\binom{n}{4}2^{-5} + 1)$ 1's. Since we have $2^{\binom{n}{2}}$ rows, we have at least $2^{\binom{n}{2}}(\binom{n}{4}2^{-5} + 1)$ 1's in the matrix, contradicting the above total.

One technical issue is how we refer to the matrix in bounded arithmetic. Given n, the number of vertices, consider the matrix in the above counting argument, except that the rows are labeled by numbers $< 2^{\binom{n}{2}}$, and the columns are labeled by numbers $< \binom{n}{4}$, both in increasing order. The row labeling naturally represents colors by considering the binary expansion of the number. The column labeling can be viewed as the $\binom{n}{4}$ K_4 subgraphs by recalling the function $\mathsf{setNumber}$ (definition 3.1.3); we associate the numbers to sets by using the inverse of $\mathsf{setNumber}$, thus using a lexicographic ordering. It would now be natural to define the matrix by a bounded formula, giving the entry at any position (a, b). However we would run into a problem in the proof: We want to be able to consider some column and find the i^{th} row with a 1 for this column. We will define

the matrix using an enumerating function $\mathbf{Ek}(c,i)$, where $c < \binom{n}{4}$ represents a K_4 subgraph, and $i < 2^{\binom{n}{2}}2^{-5}$ is an index:

$$\mathbf{Ek}(c,i) := \text{the row in which the } i^{th} \text{ 1 of column } c \text{ occurs.}$$

Ek is Σ^b_1 definable in S^1_2 so we can add it conservatively. This can be seen by describing a polynomial time (in n, not $|n|$) algorithm for this function, though we also want a formula which we can prove things about. For the unconvinced we provide a Σ^b_1 description of **Ek**:

Let $c < \binom{n}{4}$ and $i < 2^{\binom{n}{2}}2^{-5}$ be our input. The number c corresponds to a K_4 subgraph, and so gives us 6 distinct edges which correspond to 6 distinct bit positions $0 \le p_1 < \ldots < p_6 < \binom{n}{2}$ of a binary string of length $\binom{n}{2}$. Suppose the most significant of these 6 bits p_6 has k more significant bits to the left and $\binom{n}{2} - k - 1$ less significant bits to the right. Now we want **Ek** to pick out the $2^{\binom{n}{2}}2^{-5}$ colorings that make all the p_i the same color, and furthermore we want this to be done in the usual order for length $\binom{n}{2}$ binary strings. Before a formal description we give a picture of this order:

$\overbrace{0\ldots\ldots0}^{k \text{ bits}}$	$\overbrace{0}^{p_6}$	$0\ldots0$	$\overbrace{0}^{p_5}$	$0\ldots0$	$\overbrace{0}^{p_4}$	$0\ldots0$	$\overbrace{0}^{p_3}$	$0\ldots0$	$\overbrace{0}^{p_2}$	$0\ldots0$	$\overbrace{0}^{p_1}$	$0\ldots0$
							$\vdots$					
$0\ldots\ldots0$	0	$1\ldots1$	0	$1\ldots1$	0	$1\ldots1$	0	$1\ldots1$	0	$1\ldots1$	0	$1\ldots1$
$0\ldots\ldots0$	1	$0\ldots0$	1	$0\ldots0$	1	$0\ldots0$	1	$0\ldots0$	1	$0\ldots0$	1	$0\ldots0$
							$\vdots$					
$0\ldots\ldots0$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$
$0\ldots\ldots1$	0	$0\ldots0$	0	$0\ldots0$	0	$0\ldots0$	0	$0\ldots0$	0	$0\ldots0$	0	$0\ldots0$
							$\vdots$					
							$\vdots$					
$1\ldots\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$	1	$1\ldots1$

We can take i to be a binary string with $\binom{n}{2} - 5$ bits viewed as 3 segments as follows: $A_0A_1A_2$, where the A_0 are the k most significant bits, A_1 is a single bit and A_2 is a binary string with $\binom{n}{2} - k - 6$ bits. $\mathbf{Ek}(c,i)$ is a number $< 2^{\binom{n}{2}}$ which we can define by describing a binary string of the form: $A_0A_1A'_2$, where A_0 and A_1 are exactly as in string i. The A'_2 is A_2 with 5 bits inserted so they appear at positions $p_1 < \ldots < p_5$; these bits are all set equal to bit A_1.

To formalize this proof in bounded arithmetic, we mimic this counting proof, using the weak pigeonhole principle. For the ordinary probabilistic method, intuitively, the pattern was to map the set of all objects to the set of codings for the bad objects. The intuition here is to map the 1's

of matrix injectively into itself by viewing the 1's of the matrix in two ways: As row/column pairs in which the row is given first, and in which the column is given first. Now we prove theorem 3.2.2.

Proof

Supposing the claim is false we will define a Σ_1^b injection from $\binom{n}{4}2^{\binom{n}{2}}2^{-5}\left(1+\frac{2^5}{\binom{n}{4}}\right)$ to $\binom{n}{4}2^{\binom{n}{2}}2^{-5}$. This contradicts the strengthened version of the weak pigeonhole principle which follows from fWPHP, due to theorem 2.3.4, relying on the fact that n is small, so $(1+\frac{2^5}{\binom{n}{4}})$ is sufficiently large.

Now we define F. Given a number $< \binom{n}{4}2^{\binom{n}{2}}2^{-5}(1+\frac{2^5}{\binom{n}{4}}) = 2^{\binom{n}{2}}(\binom{n}{4}2^{-5}+1)$ we can see it as a pair $\langle r, c\rangle$, $r < 2^{\binom{n}{2}}$ and $c < \binom{n}{4}2^{-5}+1$. F maps $\langle r, c\rangle$ to the pair $\langle u, v\rangle$, where $u < \binom{n}{4}$ is the column in which the c^{th} 1 of row r appears, and $v < 2^{\binom{n}{2}}2^{-5}$ satisfies $\mathsf{Ek}(u, v) = r$; so the range can be taken to be $\binom{n}{4}2^{\binom{n}{2}}2^{-5}$. Note that there is such a u because our assumption guarantees enough 1's for each row; furthermore, since the row length $\binom{n}{4}$ is small, we can progress up this row till we find the proper column u. Thus we have a well-defined function. To see it is injective, suppose two distinct pairs $\langle r, c\rangle$ and $\langle r', c'\rangle$ are mapped to $\langle u, v\rangle$ and $\langle u', v'\rangle$ respectively. If $r = r'$ (the rows are the same) and $c \neq c'$ then the c^{th} 1 and the c'^{th} 1 occur in different columns, so $u \neq u'$. If $r \neq r'$ (suppose $r < r'$) then assuming $u = u'$ (otherwise we are done) $v = \mathsf{Ek}(u, r) < \mathsf{Ek}(u, r') = v'$. The map F is Σ_1^b since it only uses Ek and other Σ_1^b defined functions (in S_2^1).

□

Now we consider another theorem proved by the same method.

Theorem 3.2.3 *[4] A graph with e edges and n vertices contains a bipartite subgraph with at least* $e\frac{n+1}{2n}$ *edges.*

If the vertex set is $[n]$, then this theorem says that there is a subset $B \subseteq [n]$, so that the number of crossing edges for B (i.e. an edge is "crossing" for B if it has one vertex in B and one outside B) is at least $e\frac{n+1}{2n}$. This can be proven by a similar probabilistic argument and associated counting argument. We give the counting argument.

Proof

Let G be a graph with vertices $[n]$ and e edges. The proof comes in two cases, for n even or odd. Just consider the case of n even, so $n = 2r$ (n odd is similar).

Consider the $\binom{2r}{r} \times e$ matrix in which the columns are labeled by the e edges and the rows are labeled by the $\binom{2r}{r}$ size r subsets of $[2r]$. Put a 1 in entry (i, j) if the edge at column j is crossing for the set at row i, and a 0 otherwise. For each edge, $2\binom{2r-2}{r-1} = \frac{r}{2r-1}\binom{2r}{r}$ of the subsets are crossing for it, so that number gives the number of 1's per a column. Thus the total number of 1's in the matrix is $e\frac{r}{2r-1}\binom{2r}{r}$.

Now, assume for sake of contradiction that there is no bipartite subgraph with $e\frac{r}{2r-1} \geq e\frac{n+1}{2n}$ edges; thus any subset of the vertices has at most $e\frac{r}{2r-1} - \frac{1}{2r-1}$ crossing edges. So the number of 1's in any row of the matrix is at most this, and since we have $\binom{2r}{r}$ rows, we have at most $\binom{2r}{r}(e\frac{r}{2r-1} - \frac{1}{2r-1}) < \binom{2r}{r}e\frac{r}{2r-1}$ 1's in the matrix, contradicting the above total.

□

Notice that if we considered the matrix to have 2^n rows, one for each subset, not restricting them to be of size about $n/2$, then the proof would be almost the same as theorem 3.2.1, though the bound in theorem 3.2.3 would be weakened from $e\frac{n+1}{2n}$ to $e/2$.

To formalize this, we will code graphs in the usual way as a number $< 2^{\binom{n}{2}}$. The bipartite subgraph can just be taken to be a subset of the vertices $[n]$ and so we code it as a number $< 2^n$.

Theorem 3.2.4 $\mathrm{S}^1_2 + \mathit{fWPHP}(\Sigma^b_1)$ *proves*

$$\forall G < 2^{\binom{n}{2}} \ (G \textit{ has } \geq e \textit{ edges } \Rightarrow \exists B \subseteq [n] \textit{ number of edges from } B \textit{ to } ([n] - B) \textit{ is } \geq e\frac{n+1}{2n}).$$

The proof will be similar to the proof of theorem 3.2.2 though referring to the matrix will be a little more complicated. We will use a matrix in which the rows are labeled by numbers $< \binom{2r}{r}$, in increasing order. We associate these numbers in $[\binom{2r}{r}]$ with the sets $\{X \subseteq [2r] \mid \|X\| = r\}$, via the function setNumber (recall definition 3.1.3). There is no problem using setNumber's inverse since setNumber is injective. The columns are labeled by numbers $< e$. We now define our enumerating function for the matrix, which we call **En**:

$$\mathbf{En}(t, i) = \text{ the row in which the } i^{th} \text{ 1 of column } t \text{ occurs.}$$

En is Σ^b_1 definable in S^1_2 so we can add it conservatively. We sketch the idea. Let $D :=$

$$\{X \subseteq [2r] \mid \|X\| = r \text{ and } (u \in X \text{ iff } v \notin X)\}.$$

It suffices to define the inverse to **En**, a function $f : D \to [2\binom{2r-2}{r-1}]$. This function is defined analogously to setNumber, but is more complicated. In the case of setNumber we were given a

set $X \subseteq [2r]$ of some size and we gave an iterative definition with parameter a that began by considering $a = 0$. At each step the function considered whether or not a was in X. If a was not in X, the function added an offset and moved on to $a + 1$; otherwise it moved on to $a + 1$ without adding an offset. f is similar, except that the offset depends on where a is relative to u and v. Now in bounded arithmetic, using fWPHP we prove theorem 3.2.4.

Proof

We do the case of n even, so $n = 2r$. Assuming the theorem is false we can define an injection F from $e\binom{2r}{r}\frac{r}{2r-1}$ to $\binom{2r}{r}e\frac{r}{2r-1}(1 - (1/er))$ which violates fWPHP because n and $e \leq \binom{n}{2}$ are small. Given a number $< e\binom{2r}{r}\frac{r}{2r-1}$ we can see it as a pair $\langle t, i\rangle$, where $t < e$, and $i < \binom{2r}{r}\frac{r}{2r-1}$.

Let $a = \mathsf{En}(t, i)$. F maps $\langle t, i\rangle$ to a pair whose first entry is a. For the second entry we sum row a up to column t; this can be done because e is small. By assumption this sum is $\leq (e\frac{r}{2r-1} - \frac{1}{2r-1})$. So the entire map has a range of $\binom{2r}{r}(e\frac{r}{2r-1} - \frac{1}{2r-1}) = \binom{2r}{r}e\frac{r}{2r-1}(1 - (1/er))$. This map is Σ^b_1 since it only uses En and other Σ^b_1 defined functions (in S^1_2).

□

Recall theorem 3.0.6 about probabilistic witnessing. It can be applied to the statements in both theorem 3.2.2 and 3.2.4, so not surprisingly we get probabilistic algorithms to find the objects in question. The next section provides proofs in S^1_2 and thus polynomial time algorithms.

3.2.2 Conditional Probabilities

We apply the method of conditional probabilities to the two theorems we just discussed, in fact showing that the same theorems can be formalized in just S^1_2, without the weak pigeonhole principle. We work this out in detail for theorem 3.2.1 and then note how the approach also works for theorem 3.2.3.

Supposing a probabilistic argument shows that some object (like a coloring) exists, the method of conditional probabilities can sometimes provide an algorithm for finding such an object. Applied to theorem 3.2.1 this method finds an appropriate 2-coloring of the ground set $[n]$ in polynomial (in n) time. We successively color the edges. At a given stage we will have colored some edges in such a way that the expected number of monochromatic K_4 subgraphs (conditioned on the current partial coloring) is $\leq \binom{n}{4}2^{-5}$. We then consider the two ways to color the next edge, doing a computation to find out which one maintains the bound on the expectation; in the end we actually have a desired coloring. A key point for the algorithm is that it needs to be able to compute the expectation in

polynomial time. Corresponding to this, we will need to be able to compute an analogous weight function within S^1_2, showing that it provably has the right properties.

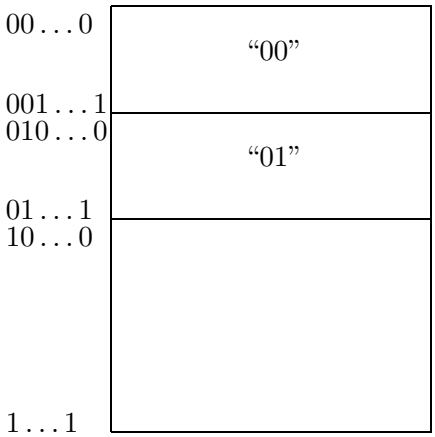
Theorem 3.2.5 S^1_2 *proves*

$$\exists C < 2^{\binom{n}{2}} \forall S \ (S \textit{ is a set of monochromatic } K_4) \Rightarrow \mathsf{size}(S) < \binom{n}{4} 2^{-5}.$$

Proof

For ease of readability we let $m = \binom{n}{2}$ (the number of edges).

To simulate the algorithm we want to be able to deal with partial colorings. It will be helpful to discuss these issues in terms of the matrix referred to in the proof of theorem 3.2.2. Recall that its rows are labeled by the colors, numbers $< 2^{\binom{n}{2}}$, viewed as binary strings of length $\binom{n}{2}$. The labels are given in increasing order, yielding the following picture:



By the partial coloring $\langle 00 \rangle$ we mean that the first 2 edges are each colored "0", where we start by coloring edges given by the most significant bits. The region of the matrix marked "00" (denoted $R_{\langle 00 \rangle}$) indicates the rows of those colorings that can extend the partial coloring $\langle 00 \rangle$. For the partial coloring $\langle 01 \rangle$ we have the corresponding region "01" and for the empty coloring $\langle\rangle$ we have the entire matrix. In general, for a partial coloring q, the associated region R_q has $2^{m-|q|}$ rows. Given a partial coloring q, the expected number of monochromatic K_4 that we get by randomly extending q is just $\sum_{t<\binom{n}{4}} \mathsf{prob}(\text{column } t \text{ has a 1 in region } R_q)$. Modeled on this we define a weight function

$$E(q) = \sum_{t<\binom{n}{4}} W(t,q),$$

where $W(t,q)$ will be a weight function for column t, under partial coloring q. We would *like to say*

$$W(t,q) := \frac{\text{number of 1's in column } t \text{ of } R_q}{2^{m-|q|}},$$

but to find the numerator directly by summing cannot be done since the sum could be long, unlike the sum used to define E (since n is small).

To get around this we will define two functions, $\mathbf{first}(t,q)$ and $\mathbf{last}(t,q)$, where t is a column and q is a partial coloring, such that the numerator will equal $\mathbf{last}(t,q) - \mathbf{first}(t,q) + 1$.

> $\mathbf{first}(t,q) :=$ the index i such that $\mathbf{Ek}(t,i)$ is the first row after the start of region R_q in which column t has a 1; if there is no such 1, then it is just $2^m 2^{-5}$.

last is defined similarly, except that we get the last row before the end of R_q, and if there is no such 1, we let it be -1 (we can make sense of a negative number). The extremal values of $2^m 2^{-5}$ and -1 for **first** and **last**, respectively, are not that important, but are chosen so that certain properties follow more smoothly (i.e. as in the properties contained in the proof of the next claim). A Π^b_1 definition of $\mathbf{first}(t,q)$ is obtained by defining it to be the smallest i such that $\mathbf{Ek}(t,i) \geq q2^{m-|q|}$. $\mathbf{last}(t,q)$ is the largest i such that $\mathbf{Ek}(t,i) < (q+1)2^{m-|q|}$. Since these functions have Π^b_1 definitions, we can conservatively add them in S^1_2. Now we can define the weight of a column under a partial coloring by:

$$W(t,q) := \frac{\mathsf{last}(t,q) - \mathsf{first}(t,q) + 1}{2^{m-|q|}}.$$

One of the key properties of the expectation is mimicked by E in the following claim (by $q \frown p$ we mean the concatenation of the 2 binary strings q and p).

Claim 3.2.6 S^1_2 *proves that for a partial coloring* q*,*

$$E(q) = \frac{E(q \frown 0) + E(q \frown 1)}{2}.$$

Proof

First we note some properties of first and last that can be checked in S^1_2:

1. $\mathsf{first}(t,q) = \mathsf{first}(t, q \frown 0)$
2. $\mathsf{last}(t,q) = \mathsf{last}(t, q \frown 1)$
3. $\mathsf{first}(t, q \frown 1) = \mathsf{last}(t, q \frown 0) + 1$

Now we give a calculation demonstrating the analogous claim for W.

$$W(t,q) = \frac{\mathsf{last}(t,q) - \mathsf{first}(t,q) + 1}{2^{m-|q|}}$$

$$= \frac{(\mathsf{last}(t, q \frown 1) - \mathsf{first}(t, q \frown 0) + 1) + (\mathsf{last}(t, q \frown 0) - \mathsf{first}(t, q \frown 1) + 1)}{2^{m-|q|}}$$

, because the second bracketed term in the numerator is equal to zero, and the first one is equal to the previous numerator (using the above properties of first and last). Continuing the calculation, the last expression equals:

$$\frac{\frac{\mathsf{last}(t, q \frown 1) - \mathsf{first}(t, q \frown 1) + 1}{2^{m-|q|-1}} + \frac{\mathsf{last}(t, q \frown 0) - \mathsf{first}(t, q \frown 0) + 1}{2^{m-|q|-1}}}{2}$$

$$= \frac{W(t, q \frown 1) + W(t, q \frown 0)}{2}.$$

Since $E(q)$ is defined in terms of sums of $W(t,q)$, a short calculation finishes the proof.

□

This claim immediately leads to the property:

$$E(q) \leq b \Rightarrow E(q \frown 0) \leq b \ \vee \ E(q \frown 1) \leq b.$$

Note that $E(\langle\rangle) = \sum_{t<\binom{n}{4}} W(t, \langle\rangle) = \binom{n}{4}\frac{2^m 2^{-5}}{2^m} = \binom{n}{4}2^{-5}$. Now using the above property we can carry out the process described before this theorem. Starting with $\langle\rangle$ we take m steps, at each step, extending our partial coloring by one more color, maintaining the bound of $\binom{n}{4}2^{-5}$ on the weight. At the end of the process we arrive at our desired total coloring q such that $E(q) \leq \binom{n}{4}2^{-5}$. Since m is small, this process can be carried out in S^1_2.

□

Theorem 3.2.2 and 3.2.5 both prove the same statement, but the more complicated proof of 3.2.5 works in a weaker theory. A similar phenomenon occurs with theorem 3.2.4.

Theorem 3.2.7 S^1_2 *proves*

$$\forall G < 2^{\binom{n}{2}} \ (G \text{ has } \geq e \text{ edges } \Rightarrow \exists B \subseteq [n] \text{ number of edges from } B \text{ to } ([n] - B) \text{ is } \geq e\frac{n+1}{2n}).$$

The proof is basically identical to that of theorem 3.2.5, though now we use En instead of Ek, redefining the functions first and last in terms of it. To find the set $B \subseteq [n]$ can be thought of as finding a 2-coloring of the vertices $[n]$. Thus the proof mimics the process of successively coloring the vertices in such a way that the weight stays *above* (rather than below) the proper bound. Another modification is to stop the process once we have r vertices colored one of the 2 colors.

How general is the this phenomenon? It is unlikely that S^1_2 proves $\mathsf{fWPHP}(\Sigma^b_1)$. However perhaps there is some kind of conservativity result.

Question 3.2.8 *Is* $S^1_2 + \mathit{fWPHP}(\Sigma^b_1)$ *conservative over* S^1_2 *under certain conditions related to linearity of expectations?*

3.3 The Local Lemma

We will discuss a theorem concerning hypergraphs and then discuss partial progress towards formalizing this in bounded arithmetic. Theorem 3.1.18 was concerned with showing certain hypergraphs are 2-colorable; we will consider another theorem of this sort. The two key parameters of theorem 3.1.18 were $m :=$ the number of hyperedges, and $k :=$ the minimum size of a hyperedge. Making m larger generally makes it harder to find a good 2-coloring, while making k larger makes this easier. Given that there is a trade-off between these two parameters, it is natural to ask what relationship between these parameters allows for a good 2-coloring. Theorem 3.1.18 answered this question by showing that $m < 2^{k-1}$ suffices. We now consider the parameter $d :=$ a bound on the number of hyperedges any given hyperedge can intersect. We ignore m and note that there is a trade-off between the parameters k and d as far as finding a 2-coloring. Finding a good trade-off between these two parameters that allows for a good 2-coloring is answered by the following theorem.

Theorem 3.3.1 *[4] Given a hypergraph in which each hyperedge has at least k elements and intersects at most d other hyperedges, if* $e(d+1) \leq 2^{k-1}$*, then the hypergraph is 2-colorable.*

The only known proof of this theorem uses the Local Lemma. The Local Lemma was proved by Erdős and Lovász [24]; we state what is referred to as the *symmetric case* of the Local Lemma.

Theorem 3.3.2 *(Local Lemma) [24] Let* $A_1, \ldots, A_m$ *be events in some probability space. Suppose each* A_i *is mutually independent of all but at most d other events* A_j*, and for each* A_i, $\mathsf{prob}(A_i) \leq p$*. If* $ep(d+1) \leq 1$ *then* $\mathsf{prob}(\bigwedge_{i=1}^n \overline{A_i}) > 0$.

Applying the Local Lemma to prove the above hypergraph theorem, our (uniform) probability space will be the set of all vertex colorings. Event A_i consists of the set of colorings which make the i^{th} hyperedge monochromatic. Two events are independent in just the case that their corresponding hyperedges do not intersect, so each event is mutually independent of all but d other events. $\mathsf{prob}(A_i) \leq \frac{1}{2^{k-1}}$ since we have at least k vertices in a hyperedge. Thus the condition of theorem 3.3.1 is exactly what is needed to apply the Local Lemma. Thus $\mathsf{prob}(\bigwedge_{i=1}^{n} \overline{A_i}) > 0$, and so there exists a coloring not contained in any of the events A_i, that is a good 2-coloring.

We now consider the formalization in bounded arithmetic. The hypergraph will be defined as before on the ground set $[n]$, using a 2-place relation symbol $\mathtt{H}$. The parameters n and k will be small as before, but we don't have to impose a restriction on m. An additional issue is that we have to bound the number of hyperedge intersections by d. We can avoid restricting d to be small by introducing a 2-place function symbol $\mathtt{G}$ which tells us which hyperedges intersect, in a way that allows us to easily talk about how many hyperedges a given hyperedge intersects. For $r < d$ and $i < m$, the value $j := \mathtt{G}(i, r)$ tells us the index of the r^{th} hyperedge intersecting hyperedge $\mathtt{H}_i$. To code that $\mathtt{G}$ is done enumerating we will let $\mathtt{G}(i, u) = \mathtt{G}(i, u+1)$ for some $u < d$; the last non-repeated value is the last number in the enumeration. So we can say that hyperedge i intersects hyperedge j with the formula $\mathsf{intersect}(i \to j)$:

$$\exists r < d\ (\mathtt{G}(i, r) = j \wedge \forall u < r\ \mathtt{G}(i, u) \neq \mathtt{G}(i, u + 1)).$$

Our definition of a hypergraph will (indirectly) make this relation symmetric. We can now say that $\mathtt{H}$ and $\mathtt{G}$ work together to give us a hypergraph with the desired parameters, where n and k are small, but m and d need not be.

Definition 3.3.3 *Let* $\mathsf{isHyperGraph}(\mathtt{H}, \mathtt{G}, m, n, k, d)$ *be:*

$$\begin{aligned}(\forall i < m\ \mathsf{size}(\mathtt{H}_i) = k) \wedge \forall x\ (\mathtt{H}(i, x) \Rightarrow x < n)\\ \wedge\ (\forall i < m\ \exists r < d\ \mathtt{G}(i, r) = \mathtt{G}(i, r + 1))\\ \wedge\ (\forall i \neq j < m\ (\mathsf{intersect}(i \to j) \Leftrightarrow \exists x < n\ \mathtt{H}(i, x) \wedge \mathtt{H}(j, x))).\end{aligned}$$

Now we can state a conjecture corresponding to theorem 3.3.1, allowing for the possibility that the formalization may involve a weaker relationship between the parameters d and k.

Conjecture 1 *For some standard rational* $r > 1$, $\mathrm{S}_2(\mathtt{H}, \mathtt{G})$ *proves that*

$$\mathsf{isHyperGraph}(\mathtt{H}, \mathtt{G}, m, n, k, d) \wedge d^r \leq 2^{k-2} \Rightarrow \exists C < 2^n \forall i < m\ \neg\mathsf{monocromaticHyper}(\mathtt{H}, C, i).$$

In fact it seems that S_2^1 plus some applications of the weak pigeonhole principle will suffice to prove this, with the parameter r not much bigger than 3. We now discuss two approaches towards proving this conjecture. The first approach is to break up the proof into two cases, depending on whether or not d is small. The second approach is to use the ideas in Alon's parallel algorithm for finding a coloring.

Approach 1: Two Cases

This approach is based on the following simple fact.

Proposition 3.3.4 *A hypergraph with m edges and n vertices, such that any hyperedge interests at most d other hyperedges, satisfies: $m \leq (d+1)n$.*

Proof

For any vertex, at most $d+1$ hyperedges contain it, otherwise we exceed d intersections. Since we have n vertices, $m \leq (d+1)n$. □

For this approach, we break up the proof into two cases: Either 1) d is small or 2) d is not small. In case (2) we will see that the ordinary Probabilistic Method can be applied, yielding only a small weakening in the d, k relationship. In case (1), if d is small, the above proposition (which we will formalize) forces m to be small too (recall n is always small). With m being small we can refer to sequences of size m, so many aspects of the usual proof of the Local Lemma can be carried out. However we do not know if the entire proof can be formalized in this case. We provide a formalization of the above proposition.

Lemma 3.3.5 $S^1_2(\mathtt{H}, \mathtt{G}) + \mathit{rWPHP}(\Pi^b_1(\mathtt{H}, \mathtt{G}))$ *proves* $\mathit{isHyperGraph}(\mathtt{H}, \mathtt{G}, m, n, k, d) \Rightarrow m \leq 2(d+1)n$.

Proof

We can give a Π^b_1 definition of an onto function h from $(d+1)n$ to m. This can immediately be turned into an injective multi-function from m to $(d+1)n$, thus implying by rWPHP that $m \leq 2(d+1)n$.

h is defined by mapping a pair $\langle x, i\rangle$, where $x < n$ and $i < d+1$, to a number $< m$. First we consider x and let $f(x) :=$ the smallest $y < m$ such that $\mathtt{H}(y, x)$ ($y = 0$ if x is in no hyperedge); this makes the definition Π^b_1. Then we consider this y. If $i = 0$, then $h(\langle x, i\rangle) := y$. Otherwise h maps $\langle x, i\rangle$ to $\mathtt{G}(y, i-1)$, the i^{th} set intersecting $\mathtt{H}_y$.

h is onto because given any $y < m$, take any vertex x such that $\mathtt{H}(y, x)$. If $f(x) = y$, then $\langle x, 0\rangle$ is mapped to y. Otherwise $\mathtt{H}_{f(x)}$ intersects $\mathtt{H}_y$, so for some $u < d$ $\mathtt{G}(f(x), u) = y$, so h maps $\langle x, u+1\rangle$ to y.

□

For this approach consider what happens if d is not small. In this case we can do an application of the ordinary probabilistic method, using theorem 3.1.19, which requires $m \leq 2^{k-2}$. Since n is

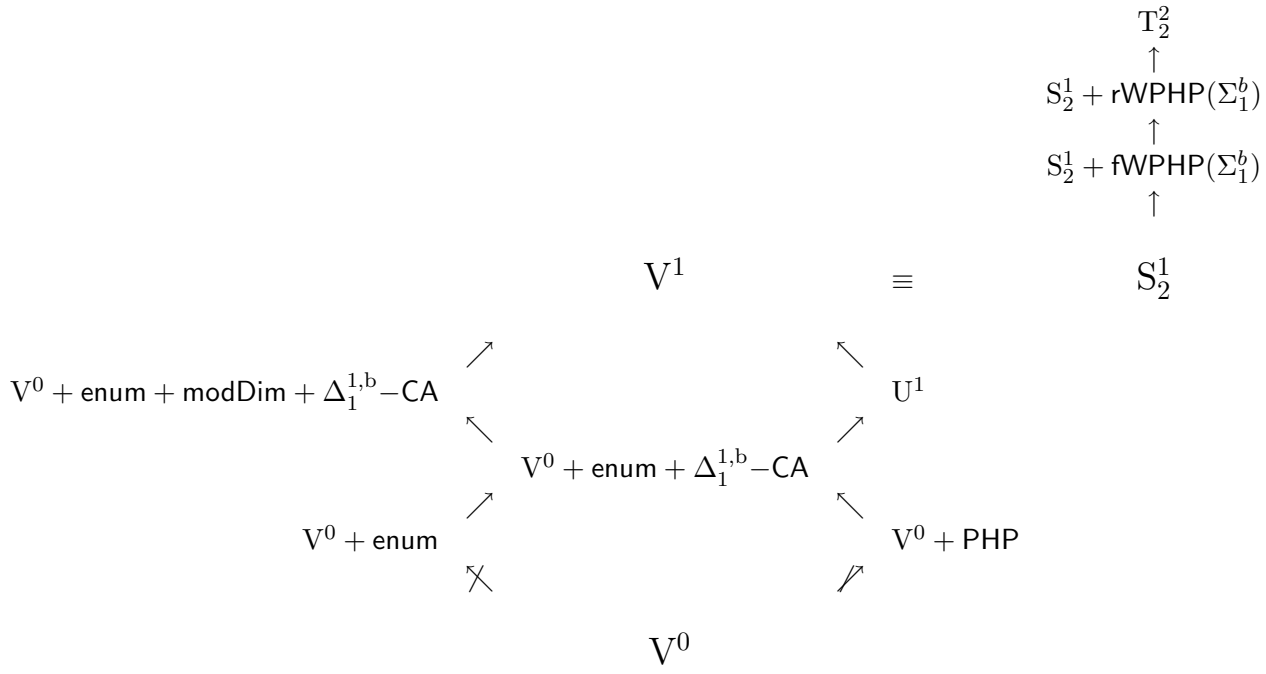
small and d is not, $n < d^{\epsilon}$ for any standard rational $\epsilon > 0$. Thus $m \leq 2(d+1)n < d^{1+\epsilon} \leq 2^{k-2}$, for any standard rational $\epsilon > 0$. So for the case of d not small we can prove the conjecture for $r := 1 + \epsilon$, working in $\mathrm{S}^1_2(\mathtt{H}, \mathtt{G}) + \mathsf{rWPHP}(\Pi^b_1(\mathtt{H}, \mathtt{G}))$. If d is small then since $m \leq 2(d+1)n$, m is also small. Given that all the parameters are small, this facilitates talking about various objects appearing in the standard proof of the Local Lemma; perhaps this would allow us to formalize this case.

Approach 2: Alon's algorithm

Another approach to the conjecture is to try formalizing an algorithm. Beck [7] found a polynomial time (in m and n) algorithm for finding a good 2-coloring of the hypergraph of theorem 3.3.1, with a weakening in the d, k relationship. Alon [2] strengthened this to a parallel algorithm which uses polynomial (in m and n) many processors, but runs in polynomial (in $\log m$) time, also suffering a weakening in the relationship between d and k. It seems very feasible that one of these could be used to prove the conjecture. In fact I believe Alon's approach will formalize. It is possible that it could be used in conjunction with the first approach to improve the d, k relationship over just using Alon's approach by itself.

Chapter 4

Second Order Bounded Arithmetic



We will develop second-order bounded arithmetic for use in the next chapter on set systems and some of the work on Ramsey theory in chapter 6. Of the various theories of this sort, we will describe (in section 4.1) Cook and Kolokolova's presentation [12] of Zambella's systems [48]. A well-known translation (presented in section 4.2) shows that a hierarchy of second order theories is isomorphic to Buss' first-order hierarchy within S_2. Thus this work can be taken to refer to Buss'

standard theories. We will in fact be interested in working in between the second order theories V^0 and V^1. V^1 is isomorphic to S^1_2 and V^0 is strictly weaker than V^1, thus we are using the second order theories as a convenient way to work below S^1_2. We choose to use the second order theories rather than the first-order theories for at least two reasons. On the one hand they provide an intuitive context for theorems that make a lot of reference to sets. Secondly, the weaker second order theory is more natural from the standpoint of complexity theory, since V^0 corresponds to the complexity class AC_0, while S^0_2 has no apparent complexity class as a friend.

Before giving the formal definitions we consider the general picture of how the various theories tie together (see the above figure). We note that "$V^1 \equiv S^1_2$," meaning the two theories are isomorphic (as discussed in section 4.2). Writing $T_1 \rightarrow T_2$ we mean that theory T_2 is at least as strong as T_1; if the arrow has a slash (i.e. "$\not\rightarrow$") this indicates T_2 is strictly stronger. Above S^1_2 we have the two kinds of WPHP schema used in the probabilistic method (chapter 3). V^1 corresponds to polynomial time reasoning and V^0 to AC_0 reasoning. In between these two theories we indicate the theories of interest to us. PHP and enum are each a single formula, referred to collectively as **counting principles** (developed in section 4.3). When added to V^0 they yield proper extensions as indicated. None of the other extensions are known to be proper, though this is believed to be the case. $\Delta^{1,b}_1\text{-}\mathsf{CA}$ is a comprehension schema and modDim is a linear algebra principle given by a single formula (discussed in the next chapter). We will not discuss U^1 in detail, just mentioning it for context. By U^1 we mean Buss' $U^1_1(\mathsf{BD})$ modified a la Zambella as $V^1_1(\mathsf{BD})$ was modified to become V^1. Thus V^0 is acting as our base theory which we add axioms to as needed.

Working over the weak base theory V^0 we will have to be careful about coding issues, so we will provide more detail in this context than has been provided for the first-order case. We develop trees in section 4.4 and some machinery for working with them over V^0. Objects ranging from sequences of numbers, to matrices, to polynomials will be coded using trees. In section 4.5 we use trees to formalize sequences of numbers and associated operations.

4.1 The Second-Order Theory

The theorems and definitions of this section mostly come from [12], which is based on Zambella's work [48]. We will now have two kinds of basic objects in our theory, numbers and sets of numbers (sets will always be taken to be finite sets of natural numbers unless otherwise stated). This second order theory will be built on top of $I\Delta_0$ by taking that theory as its first-order part, and then adding certain comprehension axioms along with a few basic facts about sets.

The language consists of $\{0, 1, +, *, 0, \mathsf{max}, \in\}$. There are two kinds of variables, number variables (denoted by lower case letters) and set variables (denoted by upper case letters). The symbols have their usual intended meaning, though max requires some explanation. Given a set X, $\mathsf{max}(X)$ is one plus the largest number in X (so $\mathsf{max}(\{0, 2, 6\}) = 7$, while $\mathsf{max}(\{\}) = 0$); not being the literal

maximum fits the empty set nicely into this definition. This highlights a key aspect of this theory: All sets are bounded, so max is well-defined. A useful abbreviation is "$X < n$", which stands for $(\forall x \in X\ x < n)$.

An alternative view of these second order theories will be to view sets as binary strings. Given a set X, such that $n = \mathsf{max}(X)$, X can be viewed as a length $(n-1)$ binary string $\omega(X)$, where bit i $(i < (n-1)\ )$ is 1 iff $i \in X$. For example, if $X = \{0, 2, 6\}$, then the corresponding string $\omega(X)$ is 000101. Note that the element $(\mathsf{max}(X) - 1) \in X$ just serves to indicate when the leading zeroes (if any) end. In referring to binary strings we say that a bit is to the **right** of another bit, if it occupies a less significant bit, and **left** if more significant, corresponding to how we wrote the example X above. In keeping with this view, all strings and sequences built on top of these sets will respect this order. We switch views of the second order objects (sets versus strings) depending on the application at hand.

As in the first order case, we have a finite set of defining axioms for the symbols of the language.

Definition 4.1.1 *Let* BASIC^2 *be* BASIC^0 *plus the following set of formulae:*

1. $0 \leq x$
2. $(x \leq y \wedge y \leq z) \Rightarrow x \leq z$
3. $x \leq y \vee y \leq x$
4. $x \leq y \Leftrightarrow x < y + 1$
5. $x \neq 0 \Rightarrow \exists y(y + 1 = x)$
6. $y \in X \Rightarrow y < \mathsf{max}(X)$
7. $y + 1 = \mathsf{max}(X) \Rightarrow y \in X$

The last two axioms are the only new basic axioms referring to sets. The other ones are in fact theorems of $\mathrm{I}\Delta_0$ (using induction). The second order theories have comprehension axioms instead of induction axioms, but induction claims will be theorems (theorem 4.1.5).

Definition 4.1.2 ***(Comprehension Axiom)*** *For a formula* ψ, $\psi{-}\mathsf{CA}$ *is:*

$$\exists X < y \forall z < y\ (z \in X \Leftrightarrow \psi(z))$$

When we use this notation it is implicit that X *does not appear free in* ψ*, though other unmentioned number and set variables may appear in* ψ*.*

Notice that in addition to the defining formula ψ, the comprehension axiom requires the bound y. If Φ is a class of formulae, then Φ−CA is the set of formulae $\{\psi\text{−CA} \mid \psi \text{ in } \Phi\}$.

In the first-order case we had induction axioms for various first-order bounded formulae. We will define analogous formulae classes for the second order context, for use in the comprehension axioms.

Definition 4.1.3 *We call the following kinds of quantifiers* ***bounded set quantifiers****. In what follows, t is a term not containing the set variable X.*

- *Let* $(\exists X < t\ \psi)$ *abbreviate* $\exists X\ (X < t \wedge \psi)$.
- *Let* $(\forall X < t\ \psi)$ *abbreviate* $\forall X\ (X < t \Rightarrow \psi)$.

By a **second-order bounded formula**, we mean a formula in which all the quantifiers are either first or second order bounded quantifiers, with free variables of both kinds allowed. We now define subclasses of the second-order bounded formulae. We define $\Sigma_i^{1,b}$ and $\Pi_i^{1,b}$ formulae exactly like the first-order Σ_i^b and Π_i^b, respectively, except that now we count the alternation of the bounded set quantifiers and ignore the bounded number quantifiers. This is a non-essential deviation from Cook's more strict definition (this is discussed further following lemma 4.1.11). For a theory T a formula ψ is $\Delta_i^{1,b}$ (in T) if ψ is in $\Sigma_i^{1,b}$ and there is a formula ϕ in $\Pi_i^{1,b}$ with the property that T proves $\psi \Leftrightarrow \phi$; if T is clear from context we do not mention it.

Note that while the names for the first-order classes of formulae, such as Σ_i^b, is quite standard, there are some different ways to refer to the second order formulae classes. We choose the above notation because of its resemblance to the first-order case. It should be noted that Buss uses the same notation for formulas defined in the same way, but in a language with the smash function #. Now we define the systems V^i introduced by Zambella (he originally called these systems $\Sigma^{\mathrm{p}}_{\mathrm{i}}$−comp).

Definition 4.1.4 *For $i \geq 0$, let V^i be the theory consisting of* BASIC^2 *and* $\Sigma^{1,\mathrm{b}}_{\mathrm{i}}$−*CA.*

Note that for $i \geq 1$, V^i is simply a nicer presentation of Buss' V^i_1(BD) in which set variables are not required to have term superscripts indicating their bound.

It is helpful to know that we can obtain induction in these theories. V^0 is strong enough to prove induction for a set variable, that is X−IND. This notation was originally used to refer to a schema, though we use it to refer to a single axiom with the set variable X in place of the a formula.

Theorem 4.1.5 V^0 *proves* X−*IND.*

The idea of the proof is to $\Sigma_0^{1,b}$ define the numbers where the inductive claim fails, and find the least one. Notice we use $\Sigma_0^{1,b}$−CA to carry out this argument. This theorem allows us to freely use Φ−IND in a theory that contains both V^0 and Φ−CA.

As in the first-order case, a version of Parikh's theorem holds for these theories. The definition of a bounded theory is exactly as before, though now we are allowed second-order bounded quantifiers in addition to the first-order bounded quantifiers.

Theorem 4.1.6 *(Parikh's Theorem for Second-Order Theories) If* T *is a second-order bounded theory that contains* V^0 *and* $\psi(\vec{x}, \vec{X}, y, Y)$ *is a second-order bounded formula such that* T *proves* $\forall \vec{x}, \vec{X}\ \exists y, Y\ \psi(\vec{x}, \vec{X}, y, Y)$*, then there is a term* t*, not containing* y *or* Y *such that* T *proves* $\forall \vec{x}, \vec{X}\ \exists y < t\ \exists Y < t\ \psi(\vec{x}, \vec{X}, y, Y)$*.*

As in the first order theories we will want to add functions or relations conservatively. As before (lemma 2.1.6), we can add definable functions and relations, noting that we can now have set parameters. It is especially useful when we can use the new function and relation symbols in the comprehension axioms (analogously for the first-order theories we wanted to use the symbols in the induction axioms). For a new function or relation symbol R, let $\Sigma_i^{1,b}(\mathtt{R})$ be defined as $\Sigma_i^{1,b}$ is, though in the language extended by R; other formulae classes are relativized similarly. By $V^i(\mathtt{R})$ we mean BASIC$^2 + \Sigma_i^{1,b}(\mathtt{R})$−CA. For a bounded theory T extending V^i we let T(R) be T + V^i(R). A new distinction for second order logic is to distinguish two kinds of functions, those with a number as output (**number-valued**) and those with a set as output (**set-valued**); otherwise we use the same definitions from first-order logic.

Theorem 4.1.7 *Let* T *be a bounded theory containing* V^0*.*

1. *If* $\psi(\vec{x}, \vec{X}, y)$ $\Sigma_0^{1,b}$ *defines a number valued function in* T*, then for a new function symbol* f T(f) + $(\psi(\vec{x}, \vec{X}, y) \Leftrightarrow \mathtt{f}(\vec{x}, \vec{X}) = y)$ *is a conservative extension of* T*.*
2. *If* $\psi(\vec{x}, \vec{X})$ *is* $\Sigma_0^{1,b}$ *and* R *is a new relation symbol, then* T(R) + $(\psi(\vec{x}, \vec{X}) \Leftrightarrow \mathtt{R}(\vec{x}, \vec{X}))$ *is a conservative extension of* T*.*

In V^0 we can add certain number-valued functions, but set-valued functions present a problem since referring to the value of the function requires a set quantifier; this means such functions cannot be used in the comprehension axioms of V^0. However special set valued functions with **bit definitions** can be added.

Definition 4.1.8 *Let* Φ *be a formula class. A set valued function* $\mathbf{f}(\vec{x}, \vec{X})$ *is* Φ ***bit definable*** *if there is a formula* $\psi(\vec{x}, \vec{X}, i)$ *in* Φ *and a term* t *such that for any* $\vec{x}$ *and* $\vec{X}$ *,*

$$i \in \mathbf{f}(\vec{x}, \vec{X}) \quad \textit{iff} \quad i < t(\vec{x}, \mathsf{max}(\vec{X})) \quad \textit{and} \quad \psi(\vec{x}, \vec{X}, i).$$

Notice that given any such formula and term, we can take it as bit defining some function; if the definition is simple enough we can add it conservatively to a relativized extension.

Lemma 4.1.9 *Let* T *be a bounded theory containing* V^0*. If* $\psi(\vec{x}, \vec{X}, y)$ *is a* $\Sigma_0^{1,b}$ *formula and* t *is a term, then for a new function symbol* $\mathtt{f}$*,* $\mathrm{T}(\mathtt{f}) + (\mathtt{f}(\vec{x}, \vec{X}) = Y \Leftrightarrow (\forall i\ i \in Y \Leftrightarrow i < t(\vec{x}, \mathsf{max}(\vec{X})) \ \wedge \psi(\vec{x}, \vec{X}, i)))$ *is a conservative extension of* T*.*

We will mainly be concerned with V^0, V^1, and theories in between. One of the main in between theories will be to consider V^0 with stronger comprehension.

Definition 4.1.10 *Let* $\Delta_{\mathrm{i}}^{1,\mathrm{b}}\text{-}\mathsf{CA}$ *be the following schema:*

$$(\forall \vec{x}, \vec{X}\ \psi(\vec{x}, \vec{X}) \Leftrightarrow \phi(\vec{x}, \vec{X})) \Rightarrow \psi\text{-}\mathsf{CA},$$

where ψ *is* $\Sigma_i^{1,b}$ *and* ϕ *is* $\Pi_i^{1,b}$*.*

Our use of this axiom will always be for the case where $i = 1$, considering the theory $\mathrm{V}^0 + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$, plus some special axioms. Note that due to theorem 4.1.5, if we are working in at least $\mathrm{V}^0 + \Delta_{\mathrm{i}}^{1,\mathrm{b}}\text{-}\mathsf{CA}$, we can use $\Delta_{\mathrm{i}}^{1,\mathrm{b}}\text{-}\mathsf{IND}$. Furthermore, we can conservatively add more functions.

Lemma 4.1.11 *Let* T *be a bounded theory that contains* V^0*. Suppose* $\mathrm{T} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ *can* $\Delta_1^{1,b}$ *define a function (number or set valued) or relation. Then the function (or relation) may be conservatively added to* $\mathrm{T} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ *and freely used in the* CA *axioms.*

We now remark on a non-essential deviation from Cook which is motivated by the $\Delta_i^{1,b}$ formulae. For him the $\Sigma_i^{1,b}$ and $\Pi_i^{1,b}$ formulae are defined more strictly to be a series of bounded set quantifiers followed by a $\Sigma_0^{1,b}$ formula. However in, V^i, for $i \geq 1$, given a $\Sigma_i^{1,b}$ (or $\Pi_i^{1,b}$) formula, we can find an equivalent one of the same complexity using the strict definition. This is not the case for $\Delta_i^{1,b}$ formulae in a theory below V^i. Thus we use the more flexible definition based on Buss' definition.

To simplify working in these theories, note that we can work with sets of tuples as opposed to just sets of numbers, because the theory is built on top of $\mathrm{I}\Delta_0$, which can code standard length sequences as single numbers. When we want to code a function as a set we will do it in the usual way as a set of tuples. We can then express that F is a function from the set X to the set Y by stating

$$\forall x \in X \exists! y \in Y\ F(x, y),$$

which we abbreviate "$F : X \to Y$". For ease of readability, we will often write $F(x) = y$, rather than the more precise $F(x, y)$. It is often important that a set parameter can be bounded in order to control the complexity of some formula. To bound a set F that codes a function such that

$F : X \to Y$, notice that the largest element in F will be the number coded by the largest pair in F, which will be bound by the code of the pair $\langle \mathsf{max}(X), \mathsf{max}(Y)\rangle$ (we assume that the pair encoding is monotone). We refer to this bounding term by $\mathsf{Fbd}(X, Y)$, so $F < \mathsf{Fbd}(X, Y)$.

We now consider the connections to complexity theory. Now our functions and relations will have number and set arguments. In keeping with standard complexity theory, our sets will be viewed as binary strings. A peculiarity is that all number arguments must be input in unary, so a number input n is not taken to have length $\log_2 n$, but actually n. Thus a function $\mathbf{F}(\vec{x}, \vec{X})$ is in **polynomial time (in the second order sense)** (denoted $\mathcal{P}$) if there is a polynomial p and a Turing machine M such that M computes $\mathbf{F}(\vec{x}, \vec{X})$ in time $p(\vec{x}, \mathsf{max}(\vec{Y}))$. We can now state the witnessing theorem for V^1.

Theorem 4.1.12 *A function is in $\mathcal{P}$ iff it is $\Sigma_1^{1,b}$ definable in V^1.*

As we shall see in section 4.2 on translating, V^1 corresponds to S_2^1, so this is basically a second-order restatement of Buss' witnessing theorem. There are similar witnessing theorems for V^i $(i > 1)$ corresponding to Buss' witnessing theorems for S_2^i. However, unique to this context is a witnessing theorem for V^0.

For V^0 we will obtain a correspondence to uniform AC_0. A relation $\mathbf{R}(X)$ of one string argument is just a set of binary strings, often called a **language** in complexity theory; we call such $\mathbf{R}$ a **string relation**. Recall that AC_0 is the set of string relations that can be recognized by a constant depth polynomial size (in the length of the input string) circuit family with unbounded fan-in. By **uniform**, we mean that the circuit family can be exhibited by a log space Turing machine M, that is, the n^{th} circuit in the family is the output when n (in unary) is input to M. When referring to AC_0 we will always mean the uniform version. We begin with a definability correspondence to AC_0.

Lemma 4.1.13 *A string relation $\mathbf{R}$ is in AC_0 iff $\mathbf{R}$ is $\Sigma_0^{1,b}$ definable.*

To obtain witnessing theorems we will extend the notion of AC_0 to functions and relations of many arguments. Based on this lemma we extend AC_0 to relations, by saying that $\mathbf{R}(\vec{x}, \vec{X})$ is in AC_0 iff $\mathbf{R}$ is $\Sigma_0^{1,b}$ definable. This definition can be tied up explicitly to a complexity class as follows.

Lemma 4.1.14 *A relation $\mathbf{R}(\vec{x}, \vec{X})$ is $\Sigma_0^{1,b}$ definable iff $\mathbf{R}$ is recognized by a log-time constant-alternation alternating Turing machine.*

So as a special case, the string relations recognized by a log-time constant-alternation alternating Turing machine are exactly the string relations in AC_0. We now extend AC_0 further, to the class of AC_0 functions FAC_0.

Definition 4.1.15 *A set valued function* $\mathbf{F}(\vec{x}, \vec{X}) = Y$ *is in* FAC_0 *if there is an* AC_0 *relation* $\mathbf{R}$ *and a polynomial* p *such that:*

$$i \in Y \text{ iff } i < p(\vec{x}, \mathsf{max}(\vec{X})) \text{ and } \mathbf{R}(i, \vec{x}, \vec{X})$$

Now we can state the witnessing theorem for V^0.

Theorem 4.1.16 *(AC_0 Witnessing) A set valued function is in* FAC_0 *iff it is* $\Sigma_1^{1,b}$ *definable in* V^0.

We will apply this witnessing theorem in the section on counting to prove a counting principle is independent of V^0. Since V^1 proves this principle, a corollary is that V^1 is strictly stronger than V^0. This fits with the intuition that two theories corresponding to genuinely different complexity classes (i.e. $\mathcal{P}$ is stronger than AC_0) should themselves genuinely be different.

4.2 Translations

We now discuss the translations between Buss' S_2^i and Zambella's V^i, for $i \geq 1$. These translations have been worked out by various people ([32], [41], [45]), using Buss' equivalent $\mathrm{V}_1^i(\mathsf{BD})$. We will sketch some of the ideas behind the translation following Takeuti's discussion ([44] and [45]). Note that the translation breaks down below S_2^1, so the nice complexity correspondence for V^0 does not carry over to S_2^0.

There is a translation that takes a formula ψ in the language of S_2^i and produces a formula ψ^H in the language of V^i. The main claim about this translation is that:

Theorem 4.2.1 *Let* $i \geq i$. S_2^i *proves* ψ *iff* V^i *proves* ψ^H.

There is also a translation taking a formula ϕ in the language of V^i to a formula ϕ^L in the language of S_2^i such that:

Theorem 4.2.2 *Let* $i \geq i$. V^i *proves* ϕ *iff* S_2^i *proves* ϕ^L.

Tieing together the two translations is the following theorem.

Theorem 4.2.3 *Let* $i \geq i$.

- S_2^i *proves* $(\psi \Leftrightarrow (\psi^H)^L)$.
- V^i *proves* $(\phi \Leftrightarrow (\phi^L)^H)$.

We now discuss roughly how these translations work. For the translation from V^i to S^i_2, we have to deal with sets in the first order context. We do this by coding the sets of V^i by numbers in S^i_2. As noted earlier (see the discussion surrounding definition 2.4.1) we can code "small" length sequences (i.e. small size sets) in S^i_2. This will be fine, since the numbers of V^i will essentially be interpreted by the small numbers of S^i_2. A set X of V^i has a number bound $\mathsf{max}(X)$ which will become small under the translation, so X can be coded by a number $s < 2^{\mathsf{max}(X)}$ in S^i_2, since we can exponentiate for small powers. The elements of the set are then checked by accessing the bits of s. In the actual syntactic translation from ϕ to ϕ^L, we apply the length operator to the free number variables and to the number variables bound by an unbounded quantifier, otherwise building up terms in the same manner. For example, if ϕ is $\exists x \forall y < (z+1)\ x + y > z$, then ϕ^L would be $\exists x \forall y < |z| + 1\ |x| + y > |z|$. This causes the bounded number quantifiers of ψ to become sharply bounded quantifiers in ψ^L; the bounded set quantifiers become regular bounded quantifiers. Thus the axioms, which depend on formula complexity are preserved by this translation.

For the translation from S^i_2 to V^i, the issue is to get around the fact that V^i does not have the smash function. To do this, the numbers of S^i_2 are coded as strings. The operations of S^i_2 are all translated to operations on strings, where intuitively, the string X is viewed as the number $\sum_{i<\mathsf{max}(X)} 2^i$, its binary expansion. So a number $x < 2^n$ of S^1_2 is translated to a string X with bound n, having 1's in places according to x. To translate the operations to the strings of V^i we just consider the example of smash. Recall that for two numbers x and y, $x\#y = 2^{|x||y|}$. Given two sets (with bounds) $A < a$ and $B < b$ we want to define "$A\#B$." Viewing A and B as their binary expansions, the length of A is just $\mathsf{max}(A)$ and length of B is $\mathsf{max}(B)$. So $A\#B$ should be $2^{\mathsf{max}(A)\mathsf{max}(B)}$, which as a binary expansion is just the set C with a single 1 at the $\mathsf{max}(A)\mathsf{max}(B)+1$ bit. The set C is easy to describe and is bound by $ab+1$, which is a term in the language.

Syntactically a term of S^i_2 becomes a term/formula pair (t, A) in V^i, where the formula A describes the set and the term t provides the bound on the set. The operations of S^i_2 are translated by describing analogous operations on these pairs. In fact once we have the above intuition on how to view strings, the definitions of the operations are virtually forced. In our example with smash, the sets A and B are given by such term/formula pairs and the set C would be defined in terms of them, following exactly our intuitive description.

4.3 Counting

We will consider adding two **counting principles**, the pigeonhole principle, and a principle which allows us to find the number of elements in a set. First we consider adding the pigeonhole principle in the second order context. Recall that we had the formula $\mathsf{PHP}^m_n(\mathtt{R})$ in the first-order context, where $\mathtt{R}$ was a 2-place relation symbol. We write $\mathsf{PHP}^m_n(F)$, where F is a set variable to mean

basically the same thing, with the 2-place relation variable F in place of the relation symbol $\mathtt{R}$. Also recall that $\mathrm{I}\Delta_0(\mathtt{R})$ does not prove $\mathsf{PHP}^{n+1}_n(\mathtt{R})$, while the provability of $\mathsf{PHP}^{2n}_n(\mathtt{R})$ is an open question. This in fact implies that V^0 does not prove $\mathsf{PHP}^{n+1}_n(F)$. To see this, consider the theory $\mathrm{I}\Delta_0(\mathsf{set})$ ($\mathrm{I}\Delta_0$ extended by set variables but no extra axioms), which cannot prove $\mathsf{PHP}^{n+1}_n(F)$. Now it suffices to show that V^0 is a conservative extension of $\mathrm{I}\Delta_0(\mathsf{set})$ for formulae with no set quantifiers. This is true because a model $\mathcal{M}$ of $\mathrm{I}\Delta_0(\mathsf{set})$ can be extended to a model of V^0 by adding sets defined by $\Sigma^{1,b}_0$ formulae (with parameters from $\mathcal{M}$).

Given the fact that we do not know the status of the weak pigeonhole principle in V^0, we will just work with the usual one in this context. Furthermore, we had two versions of the pigeonhole principle, the functional and relational version. Over V^0 they are equivalent, since from a multi-function F we can define a function F' by setting $F'(x)$ equal to the smallest y such that $F(x) = y$. Thus we will write PHP to refer to PHP^{n+1}_n with either the "r" or "f" prefix allowed. Notice we often leave out reference to the free variable F which can be taken as free or universally quantified over.

A useful consequence of PHP, which turns out to be equivalent (over V^0) is a principle we call the **injective property**, which states that having an injective function from a to b implies that $a \leq b$. We can state that a function F ($F : X \to Y$) is injective with the following $\Sigma^{1,b}_0$ formula

$$\forall a \neq b \in X \; F(x) \neq F(y),$$

which we denote $\mathsf{injective}(F)$. By "$F : X \hookrightarrow Y$" we mean that $F : X \to Y$ and $\mathsf{injective}(F)$.

We can define similar formulae expressing surjectivity and bijectivity, denoted $\mathsf{surjective}$ and $\mathsf{bijective}$; the domain X and range Y will be implicit from context.

Definition 4.3.1 *Let* $\mathsf{InjectiveProperty}$ *be:*

$$\exists F \; (F : [a] \hookrightarrow [b]) \Rightarrow a \leq b$$

Lemma 4.3.2 V^0 *proves* $\mathsf{InjectiveProperty} \Leftrightarrow \mathsf{PHP}$.

The equivalence is easy to show. Also note that PHP is outright provable in V^1, though this will be improved upon by theorem 4.3.9.

Now we consider adding counting axioms that allow us to count the number of elements in a set. In translation to first-order logic, this essentially corresponds to a function which counts the number of 1's in the binary representation of a number. Such a function is definable in S^1_2, and so we shall see that this counting can be done in the corresponding second order theory V^1. However, we will work over V^0 where adding such counting axioms is a genuine strengthening.

A common notion of counting in bounded arithmetic uses the **census function**. We will introduce this approach and then describe an equivalent method that we find more convenient to

work with, using **enumerating functions**. Given a set X a census function C for X is a function of one number argument defined for $n \leq \mathsf{max}(X)$, such that $C(n)$ is the number of $x < n$ such that $x \in X$. So $C(\mathsf{max}(X))$ is the number of elements in X. **isCensus**(C, X) says that C is the census function for the set X.

Lemma 4.3.3 **isCensus**(C, X) *is* $\Sigma_0^{1,b}$ *definable.*

Proof

Let a abbreviate $\mathsf{max}(X)$ and define $\mathsf{isCensus}(C, X)$ as follows:

$$\begin{aligned}&(C : [a+1] \to [a+1]) \ \wedge C(0) = 0 \\ &\qquad\qquad\qquad\wedge \\ &\forall u < a \ (u \in X \Rightarrow C(u+1) = C(u) + 1) \\ &\qquad\qquad\qquad\wedge \\ &\forall u < a \ (u \notin X \Rightarrow C(u+1) = C(u)).\end{aligned}$$

□

We now introduce counting via enumerating functions. An enumerating function F, for a set X, is a bijection from some number a to X, which is increasing; so $F(0)$ gives us the first element of X, $F(1)$ the second, and so on, till $F(a-1)$ gives us the a^{th} and final element of X. Let $\mathsf{CF}(F, a \to X)$ be the following $\Sigma_0^{1,b}$ formula expressing that F is such a function giving a bijection from a to X:

$$(F : [a] \to X) \wedge \mathsf{bijective}(F) \wedge \forall x \leq y < a \ F(x) \leq F(y).$$

And now we can define the **counting axioms**, which assert that for any set there is a function (census or enumerating type) that counts it.

Definition 4.3.4 ***(Counting Axioms)***

- *Let* $\mathsf{enum}(X)$ *be the axiom:* $\exists F, a \ \mathsf{CF}(F, a \to X)$
- *Let* $\mathsf{census}(X)$ *be the axiom:* $\exists C \ \mathsf{isCensus}(C, X)$

An important point about the above two axioms is that they can be stated as bounded formulae, in fact $\Sigma_1^{1,b}$ formulae. For enum, we can bound F and a in terms of $\mathsf{max}(X)$, using the term Fbd (recall the discussion following lemma 4.1.11). In fact note that from $\mathsf{enum}(X)$ with witnesses F and a, we can conclude that $a \leq \mathsf{max}(X)$; this is so because we can show by induction that $F(i) \geq i$. When we refer to one of these axioms without the parameter X, we take this to mean that X is universally quantified over. We mentioned earlier that counting in the two different ways is equivalent; by that we mean the following.

Lemma 4.3.5 V^0 *proves* $\mathsf{enum} \Leftrightarrow \mathsf{census}$

Proof

($\Rightarrow$) Given X, let F and a satisfy $\mathsf{CF}(F, a \to X)$. For $u \leq \mathsf{max}(X)$ define

$$C(u) := \begin{cases} 0 & \text{if } u \leq F(0) \\ a & \text{if } u > F(a-1) \\ i & \text{if } F(i-1) < u \leq F(i) \end{cases}$$

C has a $\Sigma_0^{1,b}$ definition and V^0 proves $\mathsf{isCensus}(C, X)$.

($\Leftarrow$) Given X, let C be a census function for it. Let $a := C(\mathsf{max}(X))$. Now we define F, a function from a to X by a $\Sigma_0^{1,b}$ definition. For $i < a$, let $F(i) :=$ the number $u \in X$ such that $C(u) = i$ and $C(u+1) = i+1$. V^0 proves $\mathsf{CF}(F, a \to X)$.

□

Since we have this equivalence between census counting and enumeration, we will not discuss the census approach any further. Our purpose for introducing it was just to provide evidence for the naturalness of this approach to counting.

An important point about counting a set is that two enumeration functions for the same set, give it the same size; in fact they must be the same function.

Lemma 4.3.6 V^0 *proves:* $\mathsf{CF}(F, a \to X) \wedge \mathsf{CF}(G, b \to X) \;\Rightarrow\; (a = b) \wedge \forall u \leq a\ F(u) = G(u)$.

Proof

Fix F,G, and X, then proceed by $\Sigma_0^{1,b}$ induction on r up to $\mathsf{max}(X)$ on the formula $\forall u \leq r\ F(u) = G(u)$. The inductive step holds because the increasing property of the enumerations forces them both to pick the same "next" element of X (if any) as r increases.

□

This lemma is another reason for our choice of counting axiom. It would be natural to consider an enumerating axiom which did not require the enumerating function to be increasing. However, in addition to losing the correspondence to the census counting, we appear to lose the property (in V^0) of being well-defined, meaning that two such counting functions for the same set could put it into bijective correspondence with different numbers. With the PHP we could prove that such functions are well-defined. However we will want to be able to use enum without the PHP.

Theorem 4.3.7 V^1 *proves* enum.

Proof

Given X, we need F and a such that $\mathsf{CF}(F, a \to X)$. Do $\Sigma_1^{1,b}$ induction on r up to $\mathsf{max}(X)$ on the formula:

$$\exists F, m\ \mathsf{CF}(F, m \to X \cap [r])$$

□

Now we will use the AC_0 witnessing theorem to show that V^0 does not prove enum. This will depend on the significant fact (shown by Håstad [28]), that **Parity** is not in AC_0, where **Parity** is the string relation that holds of exactly those strings with an odd number of 1's.

Lemma 4.3.8 V^0 *does not prove* enum.

Proof

Suppose that V^0 proves enum. Then it proves that $\forall X\ \exists a\ \exists F\ \mathsf{CF}(F, a \to X)$. Then by witnessing theorem 4.1.16 we would have an FAC_0 function that from a string X can find its number of 1's and thus its parity, contradicting the fact that **Parity** is not in AC_0.

□

We have the two counting principles enum and PHP, both provable in V^1. However we obtain the following connection. The proof essentially restates Woods' [47] proof in a different context.

Theorem 4.3.9 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}} - \mathit{CA}$ *proves PHP.*

The proof uses material developed later in this chapter so we put the proof in the appendix. There are other questions concerning the relationship between PHP and enum. Since the principles are both independent of V^0, the following question is natural.

Question 4.3.10 *What is the relationship between* enum *and PHP over* V^0*?*

At least a difference in style is that enum asserts that a certain counting function exists, while PHP asserts properties of special functions, but asserts no existence. It seems that neither implies the other over V^0.

An issue we did not explore is the possibility of interesting witnessing theorems for theorems extending V^0. Given the correspondence between V^0 and AC_0 it is natural to expect certain extensions of V^0 to correspond to AC_0 extended by special gates.

In the presence of enum we will often use $\|X\|$ to refer to the size of the set X.

4.4 Trees

We will want to code the following kinds of objects as sets: binary sequences, sequences of binary sequences, sequences of such sequences, and so on. It will be useful to code these all as trees. We will first discuss a general encoding of trees as sets, calling them **string trees**. We will then define a certain kind of tree we call a **simple tree**, which will have a structure given by a number (not a string) that codes a tree. We develop simple trees because they will sometimes allow us to work in a weaker theory.

We will use a notion of tree with a designated root, ordered children (given by the **child** function), and a finite number of nodes.

Definition 4.4.1 *We have an infinite set we call the **nodes**. A tree t is a triple $(N_t, r_t, \mathbf{child}_t)$, where N_t is a non-empty finite set of nodes, $r_\mathsf{t} \in N_\mathsf{t}$ is called the **root** of t, and $\mathbf{child}_\mathsf{t}$ is a partial function $(N_\mathsf{t} \times \mathbb{N}) \to N_\mathsf{t}$. The set of trees is defined inductively as follows:*

- *$(\{r\}, r, \emptyset)$ is a tree for any node r and the empty* **child** *function $\emptyset$.*
- *Suppose $\mathsf{t}_0, \ldots, \mathsf{t}_c$ are trees and r is a node such that $N_{\mathsf{t}_0}, \ldots, N_{\mathsf{t}_c}, \{r\}$ are mutually disjoint. Then $\mathsf{t} := (N_{\mathsf{t}_0} \cup \ldots \cup N_{\mathsf{t}_c} \cup \{r\}, r, \mathbf{child}_\mathsf{t})$ is a tree where*

$$\mathbf{child}_\mathsf{t}(n,i) := \begin{cases} r_{\mathsf{t}_i} & \text{if } n = r \\ \mathbf{child}_{\mathsf{t}_j}(n,i) & \text{if } n \in N_{\mathsf{t}_j} \end{cases}$$

*The $\mathsf{t}_0, \ldots, \mathsf{t}_c$ are called the **immediate subtrees** of t; given a tree t, we use t_i to refer to these trees. The **subtrees** of t are t itself, the immediate subtrees of t, along with together with all their subtrees.*

A **path** (of length k) in a tree is a non-empty sequence of nodes $(a_0, \ldots, a_k)$ such that a_0 is the root and for i, $0 \leq i \leq k-1$, a_{i+1} is a child of a_i. The **height** of a tree is the length of its longest path. Sometimes we refer to **labeled trees**, which just means that there is an associated function on some of the nodes assigning number or set labels. A node with no children is called a **leaf**. By the height of a tree we mean the length of the longest path in the tree. By the nodes at **level** k we mean those nodes n such that there is a path of length k ending at n.

We will be interested in coding trees where each leaf is labeled by a binary string. First we describe a string relation **isTree** with its intended meaning. Then we use a formula isTree to describe it. In order to code trees we will need delimiters, in addition to just the symbols 0 and 1. To do this, we will think of pairs of binary digits as our **basic symbols**, giving us 4 basic symbols which we will use in the following way (more details will follow):

- The length 2 string "01" is abbreviated $\underline{0}$, and is intended to stand for the single digit "0".

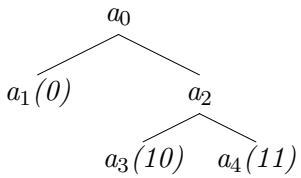
- "10" is abbreviated $\underline{1}$, intended to stand for "1".
- "00" and "11" are used as delimiters.

We will want **delimiters of order k**, for $k > 0$. We refer to such a delimiter with the symbol "$|_k$", which will be taken as a shorthand for the string "$0011\ldots1100$" where there are $2k$ consecutive 1's in the binary string, with 2 zeroes at either end; we say that the basic symbols $\underline{0}$ and $\underline{1}$ are of order 0. To facilitate referring to these basic symbols we define a function which tells us the length of a string encoding a symbol of order k:

$$\mathsf{blen}(k) := \begin{cases} 2 & \text{if } k = 0 \\ 2k+4 & \text{if } k \geq 1 \end{cases}$$

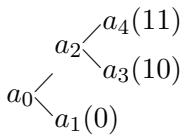
We start with an example describing a tree by a usual picture for a tree with the root at the top and children indicated by a line.

Example 4.4.2 *Let t be the following tree with leaves labeled as indicated by the binary strings in parentheses:*



This can be represented by the following binary string T: $|_2\ \underline{0}\ |_2\ |_1\ \ \underline{1}\ \underline{0}\ |_1\ \underline{1}\ \underline{1}\ |_1\ |_2$.

Notice that in keeping with our construction using sets, the children of a node, are ordered from the right being the least significant, to the most significant at the left. We may also write trees sideways, so we could write the above tree as:



Now we give precise definitions.

Definition 4.4.3 *(String Trees)*

- *An **order 0 string tree** is a non-empty finite string of elements from $\{\underline{0}, \underline{1}\}$.*

- *For $k > 0$, an **order k string tree** is a binary string of the following form:*

$$|_k\ T_{c-1}\ |_k \ldots |_k\ T_1\ |_k\ T_0\ |_k,$$

where the T_i are string trees of order 0 or order $(k-1)$.

Notice that a string tree T is essentially a particular sequence of basic symbols, where each basic symbol is in turn some binary string. We let **isTree**(T) be the string relation holding of those binary strings that are string trees. The labeled tree we intend to encode by a string tree can be recursively defined.

Definition 4.4.4 *Suppose T is a string tree of order k. We define a tree $\mathsf{t}(T)$.*

1. *If $k = 0$ then $\mathsf{t}(T)$ is a single root node labeled by the string T, replacing $\underline{0}$ by 0 and $\underline{1}$ by 1.*
2. *If $k > 0$, then T looks like:*

$$|_k\ T_{c-1}\ |_k \ldots |_k\ T_1\ |_k\ T_0\ |_k.$$

Then we construct $\mathsf{t}(T)$ by taking a new root node r with c children, where child i is the root of tree $\mathsf{t}(T_i)$.

In the example we presented, T was a string tree of order 2, such that $\mathtt{t}$ was $\mathsf{t}(T)$. It is natural to ask why we do not use the more typical encoding of trees which uses left and right brackets, which match up in the proper way. One advantage of that approach is that the size of the delimiters is fixed. However a more significant disadvantage for our purposes is the fact that using brackets seems to require counting, so the following $\Sigma_0^{1,b}$ definition would not be possible. Recall that when speaking about binary strings we use the convention that a less significant bit is to the right of a more significant bit. We use the same convention for basic symbols, which we commonly view as a single symbol, even though they are really particular binary strings.

Lemma 4.4.5 **isTree**(T) *is $\Sigma_0^{1,b}$ definable.*

Proof

First we check for the case that T is just a sequence of $\underline{0}$'s and $\underline{1}$'s in which case it is an order 0 string tree and we are done.

Otherwise, we check that it begins and ends with same order delimiter, say order k, which should be two distinct delimiters and be the highest order delimiter appearing in T. Then consider any 2 distinct delimiters d_0 and d_1 appearing in T, both of order m, with d_0 to the right of d_1. If there are no order m or higher delimiters in between d_0 and d_1, then check that we have one of the two cases:

1. We have all $\underline{0}$'s and $\underline{1}$'s in between.
2. We have two distinct order $(m-1)$ delimiters, one immediately to the left of d_0 and a different one immediately to the right of d_1.

□

Notice that the common object, the binary string is coded by a tree of order 0 (given a string tree T we can easily define its order with a $\Sigma_0^{1,b}$ formula $\mathsf{order}(T)$).

Definition 4.4.6 *Let* $\mathit{isBinarySeq}(X)$ *abbreviate* $\mathit{isTree}(X) \wedge \mathit{order}(X) = 0$.

To work with these trees it will often be convenient to isolate one of the basic symbols. For this operation, given a tree T we will simply view it as a sequence of basic symbols, ignoring its tree structure. Given some basic symbol $\underline{b}$, we let $T^{(\underline{b})}$ be exactly T with the basic symbols $\underline{b}$ (suppose they are of length r) replaced by $0 \ldots 01$, $(r-1)$ 0's followed by a single 1; the other basic symbols are replaced everywhere by 0's.

Lemma 4.4.7 $T^{(\underline{b})}$ *is* $\Sigma_0^{1,b}$ *bit-definable.*

Proof

Our bounding term is just $\mathsf{max}(T)$. Given some $x < \mathsf{max}(T)$, it will be in $T^{(\underline{b})}$ if x is the rightmost bit of some basic symbol $\underline{b}$, otherwise we leave it out.

□

A significant use of this will be that we can count the number of basic symbols $\underline{b}$ in T by counting the number of 1's in $T^{(\underline{b})}$; we call this function **symbolCount**$(T, \underline{b})$. We can define $\mathsf{symbolCount}(T, \underline{b}) = a$ by the following $\Sigma_1^{1,b}$ formula:

$$\exists F < \mathsf{Fbd}(T, T)\ \mathsf{CF}(F, a \to T^{(\underline{b})}).$$

Note that substituting the $\Sigma_0^{1,b}$ formula $T^{(\underline{b})}$ for a set variable makes sense. Furthermore, due to its complexity, V^0 can find a set B equal to $T^{(\underline{b})}$. By enum we can then find the F counting B, asserted to exist in the above definition, so $\mathrm{V}^0 + \mathsf{enum}$ shows that the above formula defines a function. It also proves that this F is unique, so the $\Sigma_1^{1,b}$ definition of $\mathsf{symbolCount}$ is equivalent (over $\mathrm{V}^0 + \mathsf{enum}$) to the following $\Pi_1^{1,b}$ definition:

$$\forall F < \mathsf{Fbd}(T, T)\ \mathsf{CF}(F, a \to T^{(\underline{b})}).$$

Thus we have a $\Delta_1^{1,b}$ definition of $\mathsf{symbolCount}$ and have proved the following lemma.

Lemma 4.4.8 **symbolCount**$(T, \underline{b})$ *is a* $\Delta_1^{1,b}$ *definable function in* $\mathrm{V}^0 + \mathsf{enum}$.

A number of proofs will follow the above style. We will want to obtain a $\Delta_1^{1,b}$ definition of a function or relation, whose definition basically consists of starting with a set, making some $\Sigma_0^{1,b}$ modifications to it, and then counting the result. In the future, for similar proofs, we will be more brief about such descriptions, though we have the above as our model of what is really going on. Another point on such definitions is that $\mathrm{V}^0 + \mathsf{enum}$ allows us to add such functions conservatively, but it *does not* allow us to use these new functions in the comprehension axioms; if we wish to use such functions freely in the comprehension axioms we need to move up to the theory $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$.

Oftentimes our trees T are best thought of as sequences, where the objects in the sequence are the immediate subtrees of T. Those children of T can in turn be thought of as sequences and so on. So given a tree T, we will refer to its number of children as its length $\mathbf{len}(T)$. We define this function for a string tree T of order $k > 0$ by counting the number of type k delimiters, and subtracting 1. For T a binary sequence, we take $\mathbf{len}(T)$ to be the total number of $\underline{0}$'s and $\underline{1}$'s. From our definition of $\mathsf{symbolCount}$ we obtain the following lemma.

Lemma 4.4.9 $\mathbf{len}(T)$ *is a* $\Delta_1^{1,b}$ *definable function in* $\mathrm{V}^0 + \mathsf{enum}$.

We will want to be able to index into trees. Given a string tree T of order k and length m, given a number $i < m$, by T_i we mean the i^{th} immediate subtree of T; it will be of order 0 or $(k-1)$. Notice that we can apply the **subscript operator** repeatedly, so $(T_i)_j$ is of order 0 or $(k-2)$.

Lemma 4.4.10 T_i *is a* $\Delta_1^{1,b}$ *definable function in* $\mathrm{V}^0 + \mathsf{enum}$.

Proof

Suppose T is of order k and $\underline{b}$ is an order k delimiter. Find $T^{(\underline{b})}$, then let F enumerate it. T_i is now essentially that part of T in between $F(i)$ and $F(i+1)$. Technically, we let T^* be the section of T intersecting $[F(i) + \mathsf{blen}(k), F(i+1) - 1]$. T_i is then $T^* - (F(i) + \mathsf{blen}(k))$.

□

Note that $\mathrm{V}^0 + \mathsf{enum}$ in fact proves that $\mathsf{isTree}(T_i)$.

Another common operation on trees will be **concatenation**, putting 2 string trees together into one string tree. Given string trees T and S, of lengths t and s respectively, both of order k, by the concatenation $T \frown S$ we mean the order k tree C of length $t+s$ in which C_i is S_i for $i < s$ and T_{i-s} for $s \leq i < t+s$. So we are thinking of T as a sequence of its t immediate subtrees and S as a sequence of its s immediate subtrees. Often we will want to concatenate while viewing one or

both of the trees as a single object. Given a string tree T of order k, we let $\langle T\rangle$ be the string tree of order $k+1$ and length 1, where $\langle T\rangle_0$ is T. This notation is general enough that it will be used on objects in a number of contexts.

One of the essential properties we will want to hold of the various operations on trees will be that they operate correctly when an extra element is appended to a tree. Many such notions will follow, though for now we just note that counting works right in this respect.

Lemma 4.4.11 $\mathrm{V}^0 + \mathsf{enum}$ *proves that for string trees* S *and* T *of order* k *and* $\underline{b}$ *a basic symbol of order* $< k$, $\mathsf{symbolCount}(T \frown S, \underline{b}) = \mathsf{symbolCount}(T, \underline{b}) + \mathsf{symbolCount}(S, \underline{b})$.

A common use of such trees will be to code a sequence of numbers. The numbers in such a sequence could be coded in binary, but unary notation is more natural; we discuss why after formalizing the ideas. It will be useful to have a coding which allows for positive and negative integers. A positive integer will be coded by a string of 1's, with an arbitrary number of leading 0's on the left. A negative integer is coded by a single 0 at the far right, followed on the left by a string of 1's which give its magnitude; it too may have leading 0's. For example +3 could be coded by 111, 0111, etc; −3 could be coded by 1110, 01110, etc. By a **unary integer**, we mean a binary string of that sort, where we use $\underline{0}$ and $\underline{1}$ for 0 and 1; we let $\mathsf{isUnaryInt}(N)$ be a $\Sigma_0^{1,b}$ formula holding of stings N that are unary integers.

Integers can also be coded by numbers, as opposed to sets. We take the pair $\langle 0, n\rangle$ to be the positive integer n, and $\langle 1, n\rangle$ to be the negative integer $-n$. In $\mathrm{I}\Delta_0$ we can prove the usual properties about these integers. There is a natural correspondence between integers as strings and integers as numbers. Given an integer $\langle s, n\rangle$ (where $s = 0$ or 1, is the sign), let $\mathbf{un}(\langle s, n\rangle)$:= the unary integer of sign s with exactly n 1's (and no leading zeroes). Let $\mathbf{num}(N)$:= the integer represented by N. The function **num** is $\Sigma_0^{1,b}$ definable in V^0 and **un** is $\Sigma_0^{1,b}$ bit definable in V^0. Note that these definitions do not count the number of 1's but rely on the unary structure of the strings, thus avoiding the use of enum. They provably do what they are supposed to.

Lemma 4.4.12

1. $\mathrm{V}^0 + \mathsf{enum}$ *proves* $\mathsf{isUnaryInt}(N) \wedge \langle s, n\rangle = \mathsf{num}(N) \Rightarrow \mathsf{symbolCount}(N, \underline{1}) = n$.
2. $\mathrm{V}^0 + \mathsf{enum}$ *proves* $N = \mathsf{un}(\langle s, n\rangle) \Rightarrow n = \mathsf{symbolCount}(N, \underline{1})$.

They are inverse to each other.

Lemma 4.4.13

1. V^0 *proves* $\mathsf{num}(\mathsf{un}(x)) = x$.

2. $\mathrm{V}^0 + \mathsf{enum}$ *proves that for a unary integer* N,
   $\mathsf{symbolCount}(N, \underline{1}) = \mathsf{symbolCount}(\mathsf{un}(\mathsf{num}(N)), \underline{1})$.

If we coded numbers in binary, notice that converting from a number to its binary representation as a set is no problem, but to go from an arbitrary binary set to a number is not possible, since the number could be exponentially large. Thus, the unary representation really is the natural way to code the numbers as sets.

Simple Trees

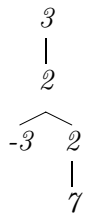
We now discuss simple trees, because we will want to be able to work with trees and sequences freely, without needing the axiom $\Delta_1^{1,\mathrm{b}}$−CA. The difference in representation will not matter when we move to stronger theories. Thus working exclusively with these simple structures does not change the issues for the stronger theories (i.e. $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}$−CA), but provides objects we can work with in the weaker theories.

Simple trees will be defined with reference to a special kind of tree we call a **structure tree**, so that given one structure tree s, we will have a number of simple trees of type s.

Definition 4.4.14 *A* ***structure tree*** *is a labeled tree with the following properties:*

- *All its nodes are labeled by positive integers except for the leaves which are labeled by an interval* (a, b)*, where* $a < b$ *are integers. As a notational convention, the labeling* $\pm a$ *means the interval* $(-a, +a)$*, and a single integer* a *means the non-negative or non-positive part of* $(-a, +a)$*, according to the sign of* a*.*
- *Given a non-leaf node labeled by the number* n*, it may have exactly 1 or* n *children.*

Example 4.4.15 *An example of a structure tree is the following tree* s*:*

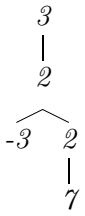


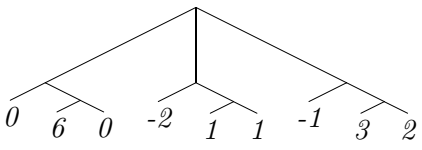
Definition 4.4.16 *(Simple Trees) For a structure tree* s*, we recursively define what we mean by a simple tree of type* s*.*

1. *If* s *is a single node labeled by an interval* (a, b)*, then a simple tree of type* s *is a single node labeled by an integer in* (a, b)*.*

2. *Otherwise a simple tree of type* s *is a tree* t *whose root* r_{t} *has* c *children, where* c *is the label of* r_{s}*, the root of* s*. Furthermore:*
   (a) *If* r_{s} *has exactly one child then each immediate subtree of* t *is a simple tree of type* s_0*.*
   (b) *If* r_{s} *has* c *children (the only other possibility for a structure tree), then each immediate subtree* t_i *is a simple tree of type* s_i

Example 4.4.17 *Using the structure trees from the previous example (we repeat it here), we give an example of a simple tree* t *of type* s*:*

$\mathsf{s} :=$ 

$\mathsf{t} :=$ 

For our choice of t in the above example, the indicated structure is forced on us, though we are free to choose the leaf labels within the proper constraints.

Given a simple tree t of type s, it will be useful to think of designating nodes of t by **addresses** in s.

Definition 4.4.18 *An* ***address*** *in a structure tree* s *is a pair of sequences* $a = \langle a_0, \ldots, a_k \rangle$ *and* $a' = \langle a'_0, \ldots, a'_k, a'_{k+1} \rangle$ *such that:*

- a' *is a path in* s *and* a *is a (possibly empty) sequence of natural numbers.*
- a_i *is less than the label at node* a'_i*.*
- *If node* a'_i *(for* $i \leq k$*) has more than one child, then node* $a'_{i+1} = \mathsf{child}(a'_i, a_i)$*.*

We say $a \lhd \mathsf{s}$*, without explicit reference to* a' *since it can be determined from* a*. If the address ends at a leaf we say that it is a* ***full address****, denoting this:* $a \trianglelefteq \mathsf{s}$*.*

An address $a \lhd \mathsf{s}$, where t is a simple tree of type s, designates a node in t by the following procedure: Start at the root node of t, then move to child a_0 of the root, and then child a_1 of that node, and so on till a ends. Keeping with the same example as above, $\langle 2, 0, 1 \rangle \trianglelefteq \mathsf{s}$ refers to the one node labeled 6 in simple tree t, while $\langle\rangle \lhd \mathsf{s}$ refers to the root of t. We can use this addressing to refer to any subtree of t. For an address $a \lhd \mathsf{s}$, we write t_a to refer to the subtree of t whose root is the node of t at address a. Though an address is meant to refer to a simple tree, it will be useful to also use

them with structure trees. For a structure tree s and $a \lhd \mathsf{s}$, we let s_a be the last node in sequence a'. For example, for address $\langle 2, 0, 1\rangle$, and s as above, s_a is the node labeled 7.

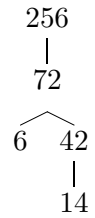
We now begin discussing how we will encode simple trees as string trees. We define a relation **isSimpleTree**(s, T), where s is a number argument intended to be a structure tree, and T is a string argument which will encode the simple tree. Just as we can code sequences in $\mathrm{I}\Delta_0$, we can code trees by numbers in $\mathrm{I}\Delta_0$. $\mathrm{I}\Delta_0$ proves basic operations on trees and can check that a tree is a structure tree. The relation **isSimpleTree** will require that s be a number which actually codes a structure tree; suppose it has height h. It will also require that **isTree**(T) and $\mathsf{order}(T) = h$. Furthermore, the tree encoded, $\mathsf{t}(T)$ (recall definition 4.4.4), must be a simple tree of type s. One may think to stop here, but in fact we want to add one more condition in order to obtain a systematic length structure. Consider any $a = \langle a_0, \ldots, a_k\rangle \lhd \mathsf{s}$ and note that we can use the previously defined subscript operator to define $T_a := (\ldots((T_{a_0})_{a_1})\ldots)_{a_k}$, which is exactly the part of T corresponding to subtree $\mathsf{t}(T)_a$. For a full address $a \trianglelefteq \mathsf{s}$, the node s_a is a a leaf which gives us an interval (u, v), so $(\mathsf{t}(T))_a$ is a number x in the interval (u, v), and T_a is a unary integer encoding of x. The final requirement in the definition of **isSimpleTree**, is that T_a must have a length independent of x, meaning that it will depend only on (u, v) and code x with leading zeroes as necessary. The absolute value of x (denoted $\mathsf{abs}(x)$), will be less than m, where m is the the larger of $\mathsf{abs}(u)$ and $\mathsf{abs}(v)$. Thus to code x we need at most m basic symbols from among $\{\underline{0}, \underline{1}\}$ (this includes room for the extra zero in the case of a negative integer); so a length $2m$ binary string will work. So we require T_a to be a binary string of length exactly $2m$ (i.e. a string of exactly m symbols from among $\{\underline{0}, \underline{1}\}$).

That finishes the definition of **isSimpleTree**; now notice some properties of it. If we fix a structure tree s, a number of different string trees T can satisfy **isSimpleTree**(s, T). However for all such T and all $a \lhd \mathsf{s}$, T_a is a binary string of a fixed length, not depending on the T in question. In fact we will define a new tree we call a **length tree** $l(\mathsf{s})$ corresponding to structure tree s, which will have the same nodes as s, with a different labeling. Since $l(\mathsf{s})$ will have the same nodes as s, given an address $a \lhd \mathsf{s}$ we can just define $l(\mathsf{s})_a$ to be node s_a in tree $l(\mathsf{s})$. $l(\mathsf{s})$ will be labeled so that the number of bits used to code T_a will be given by the label of node $l(\mathsf{s})_a$. The idea is that we know how long the bit string coding the leaves should be; we then label the rest of $l(\mathsf{s})$ recursively from the leaves up. In the recursive definition, a node $l(\mathsf{s})$ will be labeled by summing up the lengths required for the appropriate subtrees and delimiters (for delimiters recall the definition of the function $\mathsf{blen}(\cdot)$, discussed after definition 4.4.1). We define the labeling of $l(\mathsf{s})$ as follows (where h is its height):

- If a leaf node is labeled by (u, v) in s, then label this node $2m$ in $l(\mathsf{s})$, where m is the larger of $\mathsf{abs}(u)$ and $\mathsf{abs}(v)$.
- For a non-leaf node n at level m, which has one child c, label n in $l(\mathsf{s})$ by $ab + (a+1)(\mathsf{blen}(h - m))$, where a is the label of n in s and b is the label of c in $l(\mathsf{s})$.

- For a non-leaf node n at level m which has children $c_1, \ldots, c_a$, label n in $l(\mathsf{s})$ by the sum $c_1' + \ldots + c_a' + (\mathsf{blen}(h-m))(a+1)$, where c_i' is the label of c_i in $l(\mathsf{s})$.

Keeping with the same example of s above, $l(\mathsf{s})$ is the following:



We can work with structure trees and length trees in $\mathrm{I}\Delta_0$, but from a structure tree s we cannot necessarily assert the existence of $l(\mathsf{s})$ in $\mathrm{I}\Delta_0$. To see what will be needed we do a calculation. Suppose s is a number coding a structure tree of height h with n nodes labeled by numbers $< x$, so s is of size roughly x^n. $l(\mathsf{s})$ will also have n nodes, but its labels will be $< x^h$, since a node is labeled by at most the product of a path in s. Thus to code $l(\mathsf{s})$ we need a number of size about $(x^h)^n$ or roughly s^h. So if we know that s exists, we may not know that $l(\mathsf{s})$ exists. However as a convention, whenever we have a structure tree s we always assume we have the associated length tree $l(\mathsf{s})$. Also, suppose that r is the label at the root of $l(\mathsf{s})$. We always assume that we have the number r^h whenever we have the structure tree s. The point of r^h is that it will allow us to carry out certain calculations in $\mathrm{I}\Delta_0$ (used below in the "start" function). When we pass in a structure tree s as an argument to a function or relation we so not mention $l(\mathsf{s})$ or r^h, but by convention we take these to be implicit arguments; we refer to $l(s)$ and r^h as s's **associated objects**. Note that if h is standard, the preceding worries about the existence of $l(\mathsf{s})$ and r^h disappear. This will in fact be the case for our applications. The point of all this development is to allow the structure trees to be dealt with uniformly.

Given a structure tree s and an address $a \lhd \mathsf{s}$, we want to be able to find T_a without the overhead of a $\Delta_1^{1,b}$ definition (as was the case for a general string tree). To do this, suppose T is a simple tree of type s and define $\mathbf{start}(\mathsf{s}, a)$ to be the bit in T where T_a begins (recall that this does not depend on the actual T in question).

Lemma 4.4.19 $\mathbf{start}(s, a)$ *is a* Δ_0 *definable function in* $\mathrm{I}\Delta_0$.

Proof

Recall $l(\mathsf{s})$ and r^h are implicit arguments, where r is the label of the root of $l(\mathsf{s})$ and h is the height of s. Suppose $a = \langle a_0, \ldots, a_k \rangle \lhd \mathsf{s}$. We can describe the function by describing an iterative procedure starting with $i = 0$, ending with $i = k$.

1. Let $i := 0$, let $n := \mathsf{root}(\mathsf{s})$, let $u := 0$, let $m := h$.
2. If n has 1 child, then $u := u + (a_i)v + (a_i + 1)\mathsf{blen}(m)$, where $v =$ the label of n's child in $l(\mathsf{s})$.

3. Otherwise n has $> a_i$ children. Let $u := u + c_0 + \ldots + c_{a_i-1} + (a_i + 1)\mathsf{blen}(m)$, where c_i is the label at the i^{th} child of n in $l(\mathsf{s})$.
4. If $i = k$ we let $\mathsf{start}(\mathsf{s}, a) := u$, and we are done.
   Otherwise, we let $i := i + 1$ and let $m := m - 1$. We update node n as follows: *If node n has just one child, then let the new n be this child, otherwise, let the new n be child a_i of n.*
   Then goto (2).

Note that this procedure can be described by induction up to k, with u remaining $\leq r$. Since we have r^h $(k \leq h)$ as an implicit argument this is no problem. We can define the sequence of partial sums indicated in the above procedure.

□

Lemma 4.4.20 **isSimpleTree**(s, T) *is* $\Sigma_0^{1,b}$ *definable.*

Proof

Check that s is a structure tree and check its associated objects; let h be the height of s. Check that $\mathsf{isTree}(T)$. For every address $a = \langle a_0, \ldots, a_k \rangle \lhd \mathsf{s}$ $(k \geq 0)$, goto $\mathsf{start}(\mathsf{s}, a)$ in T and check that there is a delimiter immediately to the right and its order is $(h - k)$. Suppose the next order $(h - k)$ delimiter to the left begins at position b. Check that $b - \mathsf{start}(\mathsf{s}, a)$ is equal to the label at a'_{k+1} in $l(\mathsf{s})$.

□

We can very easily index into simple trees using a $\Sigma_0^{1,b}$ bit definition, without requiring that we have a counting function as was the case for the general string trees. We have defined what we mean by T_a for a simple string tree T. We will now define this in a way that allows the a to be non-standard and coincides with applying the subscript operator a standard number of times. We use the same subscript notation that was used for general string trees with the understanding that it will always be clear from context which definition we have in mind. T_a and any operations involving simple trees are always given their structure tree and its associated objects as implicit arguments.

Lemma 4.4.21 T_a *is* $\Sigma_0^{1,b}$ *bit definable in* V^0.

Proof

Suppose s is the structure tree for T and $l(\mathsf{s})$ its corresponding length tree. Let u be the label of node $l(\mathsf{s})_a$. We just say that an element x is in T_a iff $x + \mathsf{start}(\mathsf{s}, a)$ is in T and $x < u$.

□

Thus T_a can be freely used in V^0 comprehension axioms. Given a tree T, we let $T_{<k}$ be T with only its k rightmost children. For simple string trees we can give a $\Sigma_0^{1,b}$ bit definition of this operation.

We now want to describe a nice way to work with simple trees. We can typically give an easy description of a structure tree s along with its length tree, by some formula. Suppose we have another formula which tells us what the value of a leaf node should be for any full address in s. We would then like to know that there is a simple tree that corresponds to this description, essentially a kind of comprehension axiom for simple trees.

Definition 4.4.22 *Let* T *be a bounded theory containing* V^0. *Suppose* $\phi(\vec{x}, \langle \mathsf{s}, l(\mathsf{s}), d\rangle)$ *defines a function in* T *(with the arguments* $\vec{x}$ *and the tuple* $\langle \mathsf{s}, l(\mathsf{s}), d\rangle$ *as output) such that* T *proves* s *is a structure tree,* $l(\mathsf{s})$ *is its length tree, and* $d = r^h$, *where* r *is the label of the root of* $l(\mathsf{s})$ *and* h *is the height of* $l(\mathsf{s})$. *Let* $\psi(a, y; \vec{x})$ *be a formula with the parameters* $\vec{x}$ *the same as the input variables of* ϕ. *We say that* ϕ *and* ψ ***define a simple tree in T*** *if* T *proves:* $\forall a \trianglelefteq \mathsf{s}\ \exists! y \in u\ \psi(a, y)$, *where* u *is the label of* s_a *(recall that this label is an interval of integers).*

We can now describe a kind of comprehension axiom for simple trees. Note that in the following, a special case of particular interest will be if Φ is $\Sigma_0^{1,b}$, so that $V^0 + \Phi\text{-}\mathsf{CA}$ is just V^0.

Lemma 4.4.23 *(Tree Comprehension) Let* Φ *be a class of bounded formula. Suppose* $\phi(\vec{x}, \langle \mathsf{s}, l(\mathsf{s}), d\rangle)$ *and* $\psi(a, y; \vec{x})$ *define a simple tree in* $V^0 + \Phi\text{-}\mathsf{CA}$, *and* ϕ *and* ψ *are in* Φ. *Then there is a* Φ *formula* $\psi^b(i; \vec{x})$ *and a term* $t(\vec{x})$ *(in a conservative extension) such that for* $T = \{i < t \mid \psi^b(i)\}$ *(which exists in* $\Phi\text{-}\mathsf{CA}$ *)* $V^0 + \Phi\text{-}\mathsf{CA}$ *proves:*

$$\mathsf{isSimpleTree}(\mathsf{s}, T) \wedge \forall a \trianglelefteq \mathsf{s}\ (\psi(a, y) \Leftrightarrow \mathsf{num}(T_a) = y).$$

Proof

We let t be the value of the root of $l(\mathsf{s})$, which we can obtain as a term when we conservatively add a function for ϕ. t gives us a bound on the coding of T. For $\psi^b(i)$, based on $l(\mathsf{s})$, we find the longest address $a \triangleleft \mathsf{s}$ such that bit i should appear within the part coding T_a (easily done using $l(\mathsf{s})$ and the start function). If a is not full then i is in the middle of a delimiter and based on the start function and the length of a we can decide if bit i is in the part of the delimiter that is a 0 or a 1. Otherwise, i is part of a leaf node and we consider the y such that $\psi(a, y)$ and set bit i so that we indeed get the unary integer representation of y. ψ^b is indeed in Φ since it needs to refer to ψ to fill in these leaf values.

□

Note that we can apply this theorem to just go from ψ to the existence of such a T. The fact that we obtain its existence in a particular way via the definition ψ^b will be of use when we want bit definitions. Recall that a bit definition itself does not require that anything can be proved about it in a formal theory. However, the usefulness of bit definitions that we obtain by the above theorem is that we can prove (in the appropriate theory) that they correspond to some defining formula. We will not distinguish between bit definitions in the language and those in a conservative extension. In fact for our applications with standard height structure trees we can typically just give the term bound in the language.

Note an important convention about simple trees: Whenever we use simple trees we will always assume that we have their corresponding structure tree, so in particular, when we speak of a simple tree as an argument to a relation, its structure tree is an implicit argument. This convention is vital for giving $\Sigma_0^{1,b}$ definitions, but generally irrelevant to the case of giving a $\Delta_1^{1,b}$ definition (in the latter case we can freely use the $\Delta_1^{1,b}$ subscripting and length operations, though it does not hurt to have the extraneous structure tree as an argument).

4.5 Unary Arithmetic

In the last section we introduced unary integers as sets that are unary representations of numbers. We first observe that all the arithmetic on integers holds for unary integers, basically following from the fact that we have the inverse functions un and num. For example we can define the multiplication of unary integers by $\mathsf{mult}(A, B) := \mathsf{un}(\mathsf{num}(A) * \mathsf{num}(B))$. The other operations can be defined analogously. Then given any formula ψ in the language of $\mathrm{I}\Delta_0$, we can define its unary version ψ^u by replacing integers by unary integers and the symbols of ψ by their corresponding unary operation; for example, multiplication $*$ is replaced by mult and $\forall n\ \phi(n)$ is replaced by $\forall N\ (\mathsf{isUnaryInt}(N) \Rightarrow \phi(N))$. So we have:

Lemma 4.5.1 *If* $\mathrm{I}\Delta_0$ *proves* ψ*, then* V^0 *proves* ψ^u*.*

Thus in the work that follows, we will at times work with unary integers just as if they are integers, applying $+$, $*$, $=$, etc. to them, when we really mean a corresponding operation on the unary integers which are in fact sets. When we write, for example, that $X = Y$, for unary integers X and Y, we mean $\mathsf{num}(X) = \mathsf{num}(Y)$, though in fact the sets X and Y may not be the same. For the rest of this section we will be careful about this notation, but we will be more loose in later work.

The previous lemma is important, but the the real reason for introducing the unary integers is to put them into trees, going beyond $\mathrm{I}\Delta_0$. By a **vector** we mean a sequence of unary integers, coded as a simple tree.

Definition 4.5.2 *Let* $\mathsf{isVector}(V, n, a)$ *be:* $\mathsf{isSimpleTree}(n \text{—} (\pm a), V)$*.*

So $\mathsf{isVector}(V, n, a)$ says that V is a length n vector whose entries are integers between $-a$ and a. If a is replaced by $-a$ or $+a$ we mean just the non-positive or non-negative part of $(-a, +a)$, respectively. Two primary operations we will want to do to a vector $(S_1, \ldots, S_k)$ is take its sum $S_1 + \ldots + S_k$, and its product $S_1 S_2 \ldots S_k$. For products we will require that certain numbers exist, so that in fact it could be done with numbers coding sequences. However, there will be no restriction for summation and it seems necessary to use something like a counting axiom to define it. So the ultimate point of the vectors arises for summing them. Given a vector S, we let $\mathbf{sum}(S)$ be its sum, as an integer (not a unary integer).

Lemma 4.5.3 $\mathbf{sum}(S)$ *is* $\Delta_1^{1,b}$ *definable in* $\mathrm{V}^0 + \mathsf{enum}$.

Proof

By $\Sigma_0^{1,b}$ definitions we can collect together the positive integers and the negative integers, to which we can apply $\mathsf{symbolCount}$ on the $\underline{1}$'s and then take the difference.

□

Now we want to show that the definition sum provably satisfies certain basic properties. Summing a sequence agrees with usual number summation.

Lemma 4.5.4 $\mathrm{V}^0 + \mathsf{enum}$ *proves that for unary integers* A *and* B, $\mathsf{sum}(\langle A\rangle \frown \langle B\rangle) = \mathsf{num}(A) + \mathsf{num}(B)$.

Proof

We consider the case of A and B positive (others are similar).

$$\begin{array}{rcll} \mathsf{sum}(\langle A\rangle \frown \langle B\rangle) & = & \mathsf{symbolCount}(\langle A\rangle \frown \langle B\rangle, \underline{1}) & \text{, by definition of } \mathsf{sum} \\ & = & \mathsf{symbolCount}(A, \underline{1}) + \mathsf{symbolCount}(B, \underline{1}) & \text{, by lemma 4.4.11} \\ & = & \mathsf{num}(A) + \mathsf{num}(B) & \text{, by lemma 4.4.12} \end{array}$$

□

Also, summation works correctly when we append a unary integer to a vector.

Lemma 4.5.5 $\mathrm{V}^0 + \mathsf{enum}$ *proves that for a vector* S *and a unary integer* N,

$$\mathsf{sum}(S \frown \langle N\rangle) = \mathsf{sum}(\langle \mathsf{un}(\mathsf{sum}(S))\rangle \frown \langle N\rangle).$$

Proof

Again just consider the case in which we have positive unary integers.

$$\begin{aligned}
\mathsf{sum}(S \frown \langle N\rangle) &= \mathsf{symbolCount}(S, \underline{1}) + \mathsf{symbolCount}(N, \underline{1}) \\
&= \mathsf{symbolCount}(\mathsf{un}(\mathsf{symbolCount}(S, \underline{1})), \underline{1}) + \mathsf{symbolCount}(N, \underline{1}) \\
&= \mathsf{symbolCount}(\mathsf{un}(\mathsf{sum}(S)), \underline{1}) + \mathsf{symbolCount}(N, \underline{1}) \\
&= \mathsf{sum}(\langle \mathsf{un}(\mathsf{sum}(S))\rangle \frown \langle N\rangle)
\end{aligned}$$

The equalities are justified in a similar manner to the previous proof.

□

Now we consider what may seem a technical property of summing, but is actually conceptually significant, since this property is the reason that a number of proofs use PHP. If we have two vectors (of non-negative numbers) U and V such that every non-zero U_i has a unique V_j such that $U_i \leq V_j$, then $U_{m-1} + \ldots + U_0 \leq V_{k-1} + \ldots + V_0$.

Lemma 4.5.6 $\mathrm{V}^0 + PHP + enum$ *proves that if* $\mathsf{isVector}(U, m, +n)$ *and* $\mathsf{isVector}(V, k, +n)$*, and*

$$\exists F : m \to k \ ((\forall i < j < m \ (U_i \neq 0 \wedge U_j \neq 0 \Rightarrow F(i) \neq F(j))) \ \wedge \ (\forall i < m \ U_i \leq V_{F(i)}))$$

then $\mathsf{sum}(U) \leq \mathsf{sum}(V)$*.*

Proof

Let $a := \mathsf{sum}(U)$ and $b := \mathsf{sum}(V)$, exhibited by enumerating functions H and G, respectively. H injects a into the $\underline{1}$'s of U. G^{-1} injects the $\underline{1}$'s of V into b. We will define an injection I from the $\underline{1}$'s of U into the $\underline{1}$'s of V. Composing injections we have $G^{-1} \circ I \circ H : a \hookrightarrow b$, thus by PHP, $a \leq b$. Now we define I. Given a $\underline{1}$ of U, we find the i such that $\underline{1}$ is in U_i and suppose it is the r^{th} $\underline{1}$ in U_i. We map it to the r^{th} $\underline{1}$ of $V_{F(i)}$, which exists because $U_i \leq V_{F(i)}$.

□

This lemma will be used a number of times to show that different arrangements of the same numbers sum to the same total (two applications of this lemma). When applying this lemma we will frequently speak informally about finding injections (or bijections) between different ways of summing.

Now we want to deal with multiplying a sequence of unary integers. Suppose we have a sequence of m unary integers, each $< n$. Supposing we want to take the product, we will be in trouble from the start, unless we know that n^m exists; since we know we cannot define the exponentiation function,

we cannot expect to define such a product without assuming the existence of a sufficiently large number. How large a number we need will depend on whether we want to do a modular product or a regular product.

Suppose we have a sequence $a := a_{m-1}, \ldots, a_0$ such that $a_i < n$. To define the product $a_{m-1} * \ldots * a_0$, we can define a sequence of partial products $s := s_{m-1}, \ldots, s_0$, such that $s_0 = a_0$ and $s_{i+1} = a_{i+1} * s_i$, for $i < m-1$. The last number in s, s_{m-1} gives the answer. Since the numbers in this sequence can approach size n^m and we have m of them, s can be coded by a number of size about n^{m^2}. That would be fine in S^1_2, but in V^0 (built on top of $I\Delta_0$), we would have to assert the existence of n^{m^2} if we define things this way (it is conceivable that just asserting that n^m suffices).

We will actually only need modular products. Given the sequence a and the number n we want to define $a_{m-1} * \ldots * a_0 \ (\mathsf{mod}\ n)$. We could do it exactly as above, just applying $(\mathsf{mod}\ n)$ at the end, again using the existence of n^{m^2}. A more efficient approach is to redefine the sequence s by $s_{i+1} = a_{i+1} * s_i \ (\mathsf{mod}\ n)$, so the numbers in the sequence never get larger than n, so we only need to assert the existence of n^m. It is of course conceivable that V^0 could dispense with that assumption (the sequence would be input as a vector, not a number). Thus we will only take products of sequences when we know that the proper number exists and know that we have the proper bounds on our sequence. Given a vector S of m unary integers each $< n$, we let $\mathbf{product}(S, m, n)$ be its product $(\mathsf{mod}\ n)$, a number; n^m will be an implicit argument.

Lemma 4.5.7 $\mathbf{product}(S, m, n)$ *is a* $\Sigma^{1,b}_0$ *definable function in* V^0.

Proof

Since we have n^m, we can find a sequence of numbers coded by a single number s, such that s corresponds to S; by correspond we mean that V^0 proves $\forall i < m\ \mathsf{num}(S_i) = s_i$. We then define $\mathsf{product}(S, m, n)$ to be the product $(\mathsf{mod}\ n)$ of the numbers in sequence s. Since s can be coded by a number $< n^m$ it can be defined inductively in $I\Delta_0$.

□

D'Aquino and Macintyre [17] raise related number theoretic issues. We let **modular exponentiation** be the function $f(x, y, n) = x^y \ (\mathsf{mod}\ n)$. While exponentiation cannot be defined in S_2, modular exponentiation can (by repeated squaring $(\mathsf{mod}\ n)$). It is an open question as to whether or not $I\Delta_0$ can define modular exponentiation (notice that the repeated squaring approach uses sequences of length polynomial in the length of y, which is fine in S_2, but not in $I\Delta_0$). Thus to define the above product without the assumption that n^m exists would solve this open question.

When finding the product of a vector S, if the parameters m and n are apparent from context and n^m exists, we will not mention them, simply writing $\mathsf{product}(S)$. Given the definition of $\mathsf{product}$ and the correspondence between integers and unary integers, we obtain the expected properties of

product. We have similar properties for product that we have for summing. Since products are just defined by corresponding $I\Delta_0$ operations, V^0 suffices.

Lemma 4.5.8 V^0 *proves that for unary integers* A *and* B *we have* $\mathsf{product}(\langle A\rangle \frown \langle B\rangle) = \mathsf{num}(A) * \mathsf{num}(B)$.

Lemma 4.5.9 V^0 *proves that for a vector* S *and a unary integer* N*, we have:*

$$\mathsf{product}(S \frown \langle N\rangle) = \mathsf{product}(\langle \mathsf{un}(\mathsf{product}(S))\rangle \frown \langle N\rangle).$$

Chapter 5

Set Systems in Bounded Arithmetic

We have at least two motivations for this chapter on "set system theorems." Some of these theorems are used in the next chapter to give constructive Ramsey lower bounds. Furthermore, the set system theorems provide an interesting context for exploring the proof-theoretic differences between linear algebra methods and counting arguments. We will find that there is a trade-off between proof strength and theorem strength. With various linear algebra principles, we will be able to prove a number of the theorems with no loss in the strength of the theorem. Without these principles we will end up with weaker theorems.

A typical set system theorem is:

Theorem 5.0.10 *(Oddtown theorem [8]) Suppose* $\mathbf{H}$ *is a collection of subsets of* $[n]$ *in which every set in* $\mathbf{H}$ *is of odd size, and for any two distinct* A *and* B *in* $\mathbf{H}$*,* $\|A \cap B\|$ *is even. Then the number of sets in* $\mathbf{H}$ *is at most* n*.*

Given a set G, a collection $\mathbf{H}$ of non-empty subsets (pairwise distinct) of G, is called a **set system**; the set G is called the **ground set**. We will place two kinds of restrictions on set systems: limiting the allowed sizes for the sets, and limiting the allowed sizes for the intersections. Depending on the restrictions, the set system theorems will upper bound the size of $\mathbf{H}$ (i.e. the number of subsets of G that it contains).

Definition 5.0.11 *For a natural number* n*, and subsets* $K, L \subseteq [n+1]$*, we will say that a set system* $\mathbf{H}$ *is of type* (K, L, n) *if it has a size* n *ground set and:*

- *For any set* A *in* $\mathbf{H}$*,* $\|A\|$ *is in* K*.*
- *For any two distinct sets* A *and* B *in* $\mathbf{H}$*,* $\|A \cap B\|$ *is in* L*.*

In the context of this definition, we use the convention that $\overline{K}$ means the complement of K in $[n+1]$, and K (mod p) means the set of all numbers congruent (mod p) to some number in K; the conventions are the same for L. For example, the Oddtown theorem states that any type $(\{1\}$ (mod 2), $\{0\}$ (mod 2), $n)$ set system has at most n sets (equivalently, we could have called it a type $(\{1\}$ (mod 2), $\overline{\{1\}\text{ (mod 2)}}, n)$ set system).

In section 5.1 we will consider some set system theorems in the context of $V^0 + \mathsf{PHP}$ and some stronger theories which avoid any linear algebra principles. In this context, we prove a special case of the Oddtown theorem, which we will apply to a constructive Ramsey lower bound. We will then consider the "Fisher Inequality."

Theorem 5.0.12 *(The Non-Uniform Fisher Inequality [36]) For any natural number λ, a type $([n+1], \{\lambda\}, n)$ set system has at most n sets.*

We follow [5] in naming the above theorem, since it generalizes a result of Fisher, though Majumdar [36] proved the theorem as stated. The theorem is referred to as "Non-Uniform" because the sets in the set system can be of any size (i.e. the K is $[n+1] = \{0, 1, \ldots, n\}$).

Replacing the upper bound of n by $\binom{n}{\lambda+1}$, this theorem will formalize in two different theories depending on whether or not λ is standard. However, without losing the good bound, we can formalize a weaker theorem called the "Uniform-Fisher Inequality," which says that for numbers k and λ, a type $(\{k\}, \{\lambda\}, n)$ set system has at most n sets. This is referred to as "uniform" because every set must be the same size k, whereas in the Non-Uniform Fisher Inequality, the sets can be any size. We will formalize this for the case in which k and λ are standard numbers, using PHP. We then show that this claim, and other Fisher type inequalities are in fact equivalent, over V^0, to the PHP; this is an example of a "reversal."

In section 5.2 we will be able to prove stronger theorems by adding linear algebra principles to our theories. We will prove the full Non-Uniform Fisher Inequality in such a context. We will also consider what [5] refers to as the "Non-Uniform Modular Ray-Chaudhuri-Wilson theorem" (RCW theorem), named this because it modifies some work of Ray-Chaudhuri-Wilson. In chapter 6 we will apply it to prove the constructive Ramsey lower bound of Frankl and Wilson [25].

Theorem 5.0.13 *(RCW theorem [19]) Suppose $L \subseteq [n+1]$ and p is a prime. Any type $(\overline{L \text{ (mod } p)}, L \text{ (mod } p), n)$ set system has at most $\binom{n}{\|L\|} + \binom{n}{\|L\|-1} + \ldots + \binom{n}{0}$ sets.*

To work in bounded arithmetic we will use string trees to code a set system. This coding can be done in at least two natural ways, coding each set in the set system as an "incidence vector" or as a sequence of numbers. For the first approach consider a set system consisting of m subsets of the ground set $[n]$. We code each set as a length n vector with entries 0 or 1; a 1 at entry $i < n$ indicates that i is in the set and a 0 indicates that i is not in the set. The entire set system is given

by m such incidence vectors, which we can think of as an $m \times n$ matrix with 0 and 1 entries. In general we can have an $m \times n$ matrix (i.e. m rows and n columns) with entries in $(-a, a)$ for some integer a.

Definition 5.0.14 *Let* $\mathsf{isMatrix}(M, m \times n, a)$ *be:* $\mathsf{isSimpleTree}(m—n—(\pm a), M)$.

If we put $+a$ or $-a$ in place of a we mean the non-negative or non-positive part of $(-a, +a)$, respectively. The intersection of 2 sets as well as the size of a set (called size) can be easily defined.

Definition 5.0.15 *(Incidence Systems)*

- *Let* $\mathsf{isIncidenceSystem}(H, m, n)$ *be:*

$$\mathsf{isMatrix}(H, m \times n, +2) \wedge \forall i < m\ \exists x < n\ (H_i)_x = \underline{1}$$
$$\wedge\ \forall i < j < m\ \exists x < n\ ((H_i)_x \neq (H_j)_x).$$

- *Let* $\mathsf{setSizes}(H, m, n, K)$ *be:* $\forall i < m\ \exists k \in K\ \mathsf{size}(H_i) = k$.
- *Let* $\mathsf{intersections}(H, m, n, L)$ *be:* $\forall i < j < m\ \exists \lambda \in L\ \mathsf{size}(H_i \cap H_j) = \lambda$.

We can now formalize the notion of a type (K, L, n) set system with the formula

$\mathsf{isIncidenceType}(H, m, n, K, L)$:

$$\mathsf{isIncidenceSystem}(H, m, n) \wedge \mathsf{setSizes}(H, m, n, K) \wedge \mathsf{intersections}(H, m, n, L).$$

Another natural representation of a set system gives each set of the set system as a vector of non-repeating numbers from the ground set $[n]$. The set system is then the collection of such vectors. Notice that these vectors can be of varying lengths unlike the incidence vectors.

Definition 5.0.16 *Let* $\mathsf{isSetSystem}(H, m, n)$ *be:*

$$\begin{aligned}
&\mathsf{isTree}(H) \wedge \mathsf{order} = 2 \wedge \mathsf{len}(H) = m \\
&\quad \wedge \forall i < m\ \exists s < n\ (\mathsf{isVector}(H_i, s, n) \wedge \forall u < v < s\ (H_i)_u \neq (H_i)_v) \\
&\quad \wedge \forall\ i < j < m \\
&\qquad (\exists k < \mathsf{len}(H_i)\ \forall l < \mathsf{len}(H_j)\ (H_i)_k \neq (H_j)_l) \vee \\
&\qquad (\exists k < \mathsf{len}(H_j)\ \forall l < \mathsf{len}(H_i)\ (H_j)_k \neq (H_i)_l).
\end{aligned}$$

Notice that the empty sets are implicitly excluded from these set systems since we do not allow for empty vectors. We can use the same notation for intersection and set size though the subscripting actually refers to different formulae since in the later formula H is not necessarily a simple tree. We can now give an alternative formalization of a type (K, L, n) set system with the formula $\mathsf{isSetType}(H, m, n, K, L)$:

$$\mathsf{isSetSystem}(H, m, n) \wedge \mathsf{setSizes}(H, m, n, K) \wedge \mathsf{intersections}(H, m, n, L).$$

Now to say that a set system of type (K, L, n) must be of size bounded by b we write:

$$\forall H, m\ (\mathsf{isSetType}(H, m, n, K, L) \Rightarrow m \leq b),$$

sometimes leaving H and m free. Such a statement could also be made for the incidence systems. We have the following equivalence.

Proposition 5.0.17 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$ *proves:*

$$\forall H, m\ (\mathit{isIncidenceType}(H, m, n, K, L) \Rightarrow m \leq b) \Leftrightarrow \forall G, m\ (\mathit{isSetType}(G, m, n, K, L) \Rightarrow m \leq b).$$

Proof

($\Rightarrow$) Suppose G and m satisfy $\mathsf{isSetType}(G, m, n, K, L)$. It suffices to find an H such that $\mathsf{isIncidenceType}(H, m, n, K, L)$. We define H from the tree G, giving a $\Delta_1^{1,b}$ (over $\mathrm{V}^0 + \mathsf{enum}$) simple tree description. It has this complexity because we need to use the $\Delta_1^{1,b}$ subscripting into the general string tree G.

($\Leftarrow$) Suppose H and m satisfy $\mathsf{isIncidenceType}(H, m, n, K, L)$. We define G such that $\mathsf{isSetType}(G, m, n, K, L)$. We can $\Delta_1^{1,b}$ define a vector C of length m, such that C_i is the number of elements in H_i. The structure of G is then defined to be of length m, with each G_i being a vector of length C_i. We can then define the values in these vectors by subscripting into H.

□

Thus, working below $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$, as far as we know, different ways of representing the set systems are genuinely different. We choose to work with incidence systems since they allow operations like indexing (in comprehension axioms) even in the weaker systems, so they serve as a better case for analyzing the relative strength of the set system theorems.

5.1 Avoiding Linear Algebra

When working in just $\mathrm{V}^0 + \mathsf{PHP}$ a basic tool will be sunflowers.

Definition 5.1.1 *Suppose* $\mathbf{H}$ *is a set system on ground set* G*. A* ***sunflower*** *is a subset* $\mathbf{S} \subseteq \mathbf{H}$ *of size at least 2, such that there is a non-empty* $C \subseteq G$ *with the property that for any two distinct sets* A *and* B *in* $\mathbf{S}$ *we must have* $A \cap B = C$*. The set* C *is called the* ***core*** *of the sunflower. For any set* A *in* $\mathbf{S}$*, we say that the set* $A - C$ *is the* ***petal*** *of* A *(in sunflower* $\mathbf{S}$*).*

By an **isolated set** in a set system, we mean a set that does not intersect any other set of the set system.

Now we formalize a special case of the Oddtown theorem originally proved by Nagy [37]. Nagy considers a set system on $[n]$ containing sets of size 3, where any two distinct sets intersect at 0 or 2 elements; the number of sets is then at most n. As discussed in [5], Nagy's result uses linear algebra (I am not sure if Nagy's original proof in [37] is different). The known proofs of its generalization, the Oddtown theorem, use linear algebra. We provide an argument for the special case that avoids linear algebra.

Theorem 5.1.2 $\mathrm{V}^0 + PHP$ *proves* $\mathsf{isIncidenceType}(H, m, n, \{3\}, \{0, 2\}) \Rightarrow m \leq n$*.*

The informal combinatorial proof can be seen as a process of removing sets (along with their elements) from the set system, in such a way that we always remove at least as many elements as sets. First we remove any isolated sets or sets contained in just a single sunflower. Then we can partition the remaining elements of the ground set into a number of size 4 subsets so that any remaining set of the set system is fully contained in one of these size 4 subsets (by a structural argument depending on the particular parameters of the set system; to be discussed). Since there are at most $\binom{4}{3} = 4$ sets per 4 points, we are done. The formal proof uses the PHP to capture these ideas.

Proof

Let H be our set system of size m, on ground set $[n]$ with the indicated properties. We outline the approach first. We will define sets $S, N \subseteq [m]$ and corresponding sets $S^*, N^* \subseteq [n]$, such that $S \cap N = \emptyset$, $S^* \cap N^* = \emptyset$, and $S \cup N = [m]$. We will then define two injections: $I_S : S \hookrightarrow S^*$ and $I_N : N \hookrightarrow N^*$. From these injections, we can define an injection $I : m \hookrightarrow n$ by:

> Given $i < m$, I finds whether i is in S or N and uses I_S or I_N to map i into its corresponding $*-$set.

Once this is accomplished, the $\mathsf{injectiveProperty}$ implies that $m \leq n$.

We now define the sets and the injections.

- $S :=$ the set of indices $i < m$ such that set H_i is an isolated set or is contained in a single sunflower.

- $N :=$ the set of indices $i < m$ such that set i intersects another set, but is not contained in a single sunflower.

S exists since it can be $\Sigma_0^{1,b}$ defined; we can give a natural definition referring to the existence of a core with a bounded number quantifier since a core only has size 2. N is just S's complement in $[m]$. $S^* \subseteq [n]$ is defined to be the elements $x < n$ in the ground set such that $x \in H_i$ for some $i \in S$. N^* is defined similarly from N. S^* and N^* are both $\Sigma_0^{1,b}$ defined. They are disjoint since $x \in S^*$ implies $x \notin N^*$:

> Suppose $x \in S^*$, so $x \in H_i$ for some $i \in S$. If no other set contains x, then $x \notin N^*$, otherwise suppose $x \in H_i \cap H_j$ for $i \neq j$. Let $H_i = \{x, y, z\}$ and $H_j = \{x, y, w\}$. Any other set H_k containing x cannot contain z, since then $i \notin S$, and so $x, y \in H_k$ (since intersections must be 0 or 2, not 1). Thus x is only contained in sets which are part of a single sunflower with core $\{x, y\}$, i.e. $x \notin N^*$.

To define I_S consider any $i \in S$, and the associated set H_i. If H_i is isolated, I_S maps i to the smallest element of H_i. Otherwise H_i is contained in a single sunflower and the single element x in the petal of H_i is not in any other set of the set system. I_S maps i to this x. By construction, I_S is injective.

Now we define I_N. Given $i \in N$, $H_i = \{a, b, c\}$ is not in a single sunflower, meaning we have j, k (all of i, j, k are distinct) such that H_j and H_k intersect H_i, but not at the same 2 points. Suppose that $H_i \cap H_j = \{a, b\}$, and $H_i \cap H_k = \{b, c\}$. So $H_j \cap H_k$ include b and some fourth point, say $d \notin H_i$. There is possibly a fourth set $\{a, c, d\}$, but otherwise no other sets of the set system contain one of the 4 elements a, b, c, or d (we can run through the cases, considering what happens if a set contains any one of these elements, using the fact that once two sets intersect, they must intersect at precisely two elements). We simply order these 3 or 4 sets lexicographically, sending them in that order to the 4 points, so that it is injective.

□

The only known proof of the Oddtown theorem uses linear algebra. The above special case of this theorem avoids linear algebra.

Question 5.1.3 *What stronger special cases of the Oddtown theorem can be proved without linear algebra? Does the proof of theorem 5.1.2 generalize?*

We now consider the various Fisher Inequalities. For the Non-Uniform Fisher Inequality there is a bound of n on the number of sets in the set system. A simple argument gives a weakened bound of $\binom{n}{\lambda+1}$ for standard λ. This bound is obtained by noting that for every size $\lambda+1$ set, there is at most one set in the set system containing it.

Proposition 5.1.4 *For any standard λ, $\mathrm{V}^0 + PHP$ proves that:*

$$\mathsf{isIncidenceType}(H, m, n, [n+1], \{\lambda\}) \Rightarrow m \leq \binom{n}{\lambda+1}.$$

Proof

First we rule out the case in which one of the sets C in H contains only λ many elements. In this case H is just a sunflower with core C. We can define an injection $I : m \hookrightarrow n$, implying $m \leq n$. I is defined for $i < m$, by considering H_i: If H_i is C then map i to the smallest element in C; otherwise H_i's petal does not intersect any other sets of the set system so we map i to the smallest element in the petal.

Otherwise all the sets have at least $\lambda + 1$ elements; we inject $m \hookrightarrow \binom{n}{\lambda+1}$. Given $i < m$ we just consider the $\lambda + 1$ smallest numbers in set H_i, and code them into a number. Since two different sets intersect at exactly λ places this is injective, so $m \leq \binom{n}{\lambda+1}$. Notice that all these operations are easy because λ is standard.

□

We discuss how to deal with the case of λ non-standard at the end, since it gets technical and involves a stronger theory.

We can improve this to the optimal bound of n for the Uniform Fisher Inequality (with standard parameters).

Theorem 5.1.5 *For any standard natural numbers k and λ, $\mathrm{V}^0 + PHP$ proves:*

$$\mathsf{isIncidenceType}(H, m, n, \{k\}, \{\lambda\}) \Rightarrow m \leq n.$$

Proof

Note that counting sets of size k and λ can be done without enum since these parameters are standard. Suppose we have an appropriate H. We want to find a maximal sunflower in H; that is a collection of indices S in $[m]$ such that $\{H_i \mid i \in S\}$ is a sunflower and it would no longer be a sunflower if we added any other set of H to S. We cannot find a sunflower in the natural way, extending by sets until we can extend no further, since natural ways to formalize that reasoning involve $\Sigma_1^{1,\mathrm{b}}$−IND. Instead, fix any set H_i of H and any set of λ many points in H_i. Then $\Sigma_0^{1,b}$ define the set of indices that contain these λ many points. Let S be the resulting maximal sunflower. We now consider 2 cases.

For the first case, suppose $S = [m]$, so it includes all the sets of the set system. We can inject $I : S \hookrightarrow n$, thus showing $m \leq n$. Given $i < m$, if H_i is exactly the λ elements of the core, then just define I to map this i to the smallest element in the core. Otherwise i corresponds to a set H_i which contains these λ core elements and at least one other element in its petal; let I map i to the smallest element in the petal. This map is injective since petals do not overlap.

For the second case, suppose otherwise, so there is a set H_j not associated with the sunflower. Thus H_j does not contain all of the core of S, and so for every set in S, H_j must also intersect its petal (in order to get the required λ many intersections). Since the petals are disjoint and H_j contains k elements, there can be at most k sets in S. Let $E \subseteq [n]$ be the elements of S, that is $E := \{x < n \mid x \in H_i \text{ for some } i \in S\}$. E has at most k^2 elements in it. Now we consider the sets H_i such that $i \notin S$; such sets must contain some points of E, but may contain some outside points also. Consider any two distinct such sets A and B. As far as their intersection with E, there are 2^{k^2} possibilities for each, but in fact we cannot have $A \cap E = B \cap E$:

> Both A and B must contain at least $\lambda + 1$ many elements of E, since they have to intersect every set of S in λ many places but cannot do so in the most efficient way by intersecting the elements of the core. Since A and B can only intersect at λ many places, $A \cap E \neq B \cap E$.

Thus we have at most 2^{k^2} many sets not in S. Now rest of the proof is the key place we use the standardness of k (in the earlier part of the proof we only used the standardness to avoid enum, but here we use it to get the proper bound). The total number of sets is then bound by $k + 2^{k^2} \leq n$, for sufficiently large n. For smaller n we can just do the standard proof in a brute force manner, since the parameters are standard.

□

Now we prove a reversal.

Theorem 5.1.6 *For any standard $\lambda \geq 1$,*

$$\mathrm{V}^0 + \forall H, m\ (\mathit{isIncidenceType}(H, m, n, \{\lambda + 1\}, \{\lambda\}) \Rightarrow m \leq n) \text{ proves } \mathit{PHP}.$$

Proof

Suppose for contradiction that PHP does not hold, so we have F, an injection from $n+1$ to n. We amplify the F to an injection from $n + 1 + \lambda$ to n:

> Define injection $G_1 : (n + 1 + 1) \to n + 1$ by $G_1(x) := F(x)$ for $x < n + 1$, and $G_1(n + 1) := n$. Define injection $F_1 : (n + 1 + 1) \to n$ by $F \circ G_1$. Then

we obtain injection $F_2 : (n+1+2) \to n$ in a similar fashion. We can repeat this standardly many times to obtain injection $F_\lambda : (n+1+\lambda) \to n$.

We can now construct a set system H violating our set system axiom. The set system H will have $n+1+\lambda$ sets on ground set $[n+\lambda]$. For $i < n+1+\lambda$, and $x < n+\lambda$ we define

$$(H_i)_x := \begin{cases} \underline{1}, & \text{if } x = F_\lambda(i) + \lambda \text{ or } x < \lambda \\ \underline{0}, & \text{otherwise} \end{cases}$$

Every set H_i has the λ elements $\{0, 1, \ldots, \lambda-1\}$ in addition to one more unique element. Thus any 2 distinct sets intersect at λ elements and every set contains $\lambda+1$ many elements (we can enumerate these elements without use of enum). H is an appropriate set system, but the number of sets, $n+1+\lambda$, is larger than the ground set, with $n+\lambda$ elements, so we have contradicted the set system axiom.

□

Using a stronger theory, we now prove a stronger version of the statement in proposition 5.1.4, no longer requiring λ to be standard.

Theorem 5.1.7 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$ *proves that if* $\binom{n}{\lambda+1}$ *exists and* $\lambda < n/2$ *then*

$$\mathit{isIncidenceType}(H, m, n, [n+1], \{\lambda\}) \Rightarrow m \leq \binom{n}{\lambda+1}.$$

We basically want to follow the proof of proposition 5.1.4. In that proof we assigned a size $\lambda+1$ subset $S \subseteq [n]$, to a number less than $\binom{n}{\lambda+1}$. This can be done by a standard number of operations for λ standard, but when λ is non-standard, we will use the $\mathsf{setNumber}$ function of definition 3.1.3. Recall that for $S \subseteq [n]$ of size k, $\mathsf{setNumber}(S; n)$ assigned S to a number $< \binom{n}{k}$, according to its lexicographic ordering. We will actually make a few modifications to $\mathsf{setNumber}$, keeping the same name. A minor change is that $\mathsf{setNumber}$ will take a set as input rather than a number coding a set. The more significant change is that we will also input the parameter k, passing in $\binom{n}{k}$ as an implicit argument and requiring that $k \leq n/2$. We could define this function in V^1, mimicking our previous S_2^1 definition, however we want to work in a weaker theory. The requirement that $k \leq n/2$ allows for the applications of lemma 2.4.5 which shows that $\binom{n}{j}$ also exists for $j < k$ (if we are in a context where we already know that these choose values exist, we can dispense with the assumption that $k \leq n/2$, as will be the case for the RCW theorem). We can work in a weaker theory by giving a slightly modified definition that works more in parallel. To motivate the reader, this technical development will also be used in the proof of the RCW theorem and its application to the Frankl and Wilson Ramsey lower bound.

The redefinition of $\mathsf{setNumber}$ here follows basically the same idea as before. Given a set S, we think of starting a process at the least significant right side. If we see a zero at position i of S, we

define a number which counts the number of size k subsets that look like S for bits to the right of i, but have a 1 at position i.

Lemma 5.1.8 setNumber *is* $\Delta_1^{1,b}$ *definable in* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$.

Proof

Suppose we are given numbers n and k such that $\binom{n}{k}$ exists and $k \leq n/2$, along with $S \subseteq [n]$. Define a corresponding length n vector S' such that

$$S'_i := \begin{cases} \binom{n-i-1}{k-1-\mathsf{size}(S_{<i})} & \text{if } S_i = 0 \text{ and } \mathsf{size}(S_{<i}) < k \\ 0 & \text{otherwise} \end{cases}$$

Note that since $k \leq n/2$, lemma 2.4.5 guarantees the existence of the various choose functions. Then we let $\mathsf{setNumber}(S; n, k) := \mathsf{sum}(S')$.

□

Now a key point is that we can prove in the same theory that setNumber is injective on sets of the appropriate size.

Lemma 5.1.9 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$ *proves that* setNumber *is injective (i.e. for any* n *and* k *and any distinct size* k *sets* $A, B \subseteq [n]$, $\mathsf{setNumber}(A; n, k) \neq \mathsf{setNumber}(B; n, k)$*).*

Proof

Suppose we have A and B as indicated above, such that A and B are the same restricted to numbers $< i$, but $i \in A$ and $i \notin B$. Let A' and B' be the corresponding vectors defined in the above proof. Let $m := \mathsf{size}(A_{<i}) = \mathsf{size}(B_{<i})$, and $s := A'_{i-1} + \ldots + A'_0 = B'_{i-1} + \ldots + B'_0$. So $\mathsf{setNumber}(A) = A'_{n-1} + \ldots + A'_i + s$ and $\mathsf{setNumber}(B) = B'_{n-1} + \ldots + B'_i + s$. It suffices to show that $A'_{n-1} + \ldots + A'_{i+1} < B'_i = \binom{n-1-i}{k-1-m}$; the last equality is by definition and note that we can ignore $A'_i = 0$. The values of the $n - i - 1$ numbers $A'_{n-1}, \ldots, A'_{i+1}$ are at least 0 at the remaining $k - (m+1)$ places that A has a 1, so at most $n - i - 1 - (k - (m+1)) = n - i - k + m$ of the numbers from $A'_{n-1}, \ldots, A'_{i+1}$ are non-zero. The case in which the sum is the largest is if the $n - i - k + m$ entries $A'_{i+(n-i-k+m)}, \ldots, A'_{i+1}$ are non-zero. To know this in the system, we take our sum and inject it into this sum, using lemma 4.5.6. The last sum is

$$\binom{n-i-(n-i-k+m+1)}{k-2-m} + \ldots + \binom{n-i-2}{k-2-m},$$

which is less than $\binom{n-i-1}{k-1-m}$ as desired. This last point is true because of the following general point.

Claim 5.1.10

$$\binom{b-1}{b-1} + \ldots + \binom{a-1}{b-1} = \binom{a}{b}$$

We prove this by induction on a, though a quick way to see its truth is to see that both sides of the equation count the number of ways to choose a size b subset from $[a]$; the left hand side does this by fist choosing the largest element and then choosing $b-1$ smaller elements. Our desired inequality follows immediately by applying this claim for $a = n-i-1$ and $b = k-1-m$.

□

Since setNumber is injective and onto we can use this function or its inverse. Now, as mentioned earlier, to prove theorem 5.1.7, we just follow the proof of proposition 5.1.4, using setNumber, which uses a move to a stronger theory.

It is natural now to consider two general ways these results could be improved upon. Firstly, we could prove better bounds, perhaps strengthening the axioms. Secondly, we could find proofs with weaker axioms, perhaps accompanied by a worsening of the bound. The second line of thought is limited by the above reversal, theorem 5.1.6; in fact notice that an immediate consequence of the reversal is that the set system statement of theorem 5.1.5 (for $k = \lambda + 1$) is equivalent to PHP over V^0, so cannot be proved (for all parameters k and λ) in just V^0. It seems feasible to extend the reversal.

Question 5.1.11 *To what extent can the reversal of theorem 5.1.6 be extended?*

We mention two possibly promising lines of thought. De Bruijn and Erdős [18] proved a special case of the Non-Uniform Fisher Inequality.

Theorem 5.1.12 *[18] A type $([n+1], \{1\}, n)$ set system has at most n sets.*

Their proof avoids linear algebra, but a difficulty with formalizing it is a key part in which they take the sum of a sequence of rationals, which presents a difficulty beyond just summing a sequence of integers (we discuss this in more detail in the next section following definition 5.2.6). However perhaps their argument could be modified in a way that it could be formalized.

It would also be nice to improve proposition 5.1.4 or theorem 5.1.7, since they only obtain the bound of $\binom{n}{\lambda+1}$, while we in fact know that the much better bound of n (with no λ dependence) works. Consider an approach. Let b_λ be a bound on the size of a type $([n+1], \{\lambda\}, n)$ set system. An easy argument shows that $b_\lambda \leq n b_{\lambda-1}$:

> Given a set system $\mathbf{H}$ of type $([n+1], \{\lambda\}, n)$, for each element $x \in [n]$, we can define $\mathbf{H}_x$ to be those sets of $\mathbf{H}$ which contain x. Each $\mathbf{H}_x$ is a type $([n], \{\lambda - 1\}, n-1)$ set system, so there are at most $nb_{\lambda-1}$ sets in $\mathbf{H}$.

Given this fact we can start with $b_1 = n^2$ and conclude by an inductive argument (possibly outside the formal theory, so λ would need to be standard) that $b_\lambda \leq n^{\lambda+1}$. This does not improve proposition 5.1.4, although we would obtain an improvement with a sufficiently improved bound such as $b_\lambda \leq \sqrt{n}b_{\lambda-1}$.

5.2 Using Linear Algebra

Now we consider bounds obtained from linear algebraic arguments. A debt is owed to the well (and only partially) written book by Babai and Frankl, "Linear Algebra Methods In Combinatorics" [5], which collects together these results. The basic idea of this approach is to map the sets in the set system $\mathbf{H}$ to vectors in some well chosen vector space. Then we show that these vectors are linearly independent, which implies that the number of vectors, and consequently, the number of sets in $\mathbf{H}$, is bounded by the dimension of the vector space. To formalize these arguments we will want to be able to work with vectors, taking linear combinations and showing linear independence, thus we will often work in $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$. However, it still appears that we cannot prove the following linear algebra principle, which we call the **dimension principle**:

> A set of linearly independent vectors in a vector space of dimension d, can be of size at most d.

Thus, we will also add axioms that formalize such a linear algebra principle. This principle is basically inspired by the "hard matrix identities" of Soltys and Cook [43]. We will develop this material formally in section 5.2.1. Using that material, section 5.2.2 will formalize the full Non-Uniform Fisher Inequality. Section 5.2.3 will formalize the RCW theorem, using the material of section 5.2.1 and a formalization of polynomials.

5.2.1 Formalizing Linear Algebra

Soltys and Cook [43] developed a formal theory for feasible matrix operations which can be interpreted in S_2^1. Their theory has as basic objects the matrices, the field elements, and the natural numbers. Their main motivation was to analyze the complexity of various matrix theorems. Since our aim is to use linear algebra in application to combinatorics, working directly in a usual theory of bounded arithmetic seems more natural. The linear algebra we develop will correspond roughly to Soltys and Cook's theory LA.

We will begin by showing how to work with vectors and matrices over the unary integers. They will then be applied in two cases: 1) The case in which the intended field is the rationals, and 2) The case in which the intended field is $\mathbb{F}_p$, the integers modulo some prime p.

The basic operation on two length m vectors U and V will be the dot product:

$$U \cdot V = \sum_{i<m} (U_i)(V_i).$$

Lemma 5.2.1 *$(U \cdot V)$ is $\Delta_1^{1,b}$ definable in $\mathrm{V}^0 + \mathsf{enum}$.*

Proof

Let $m := \mathsf{len}(U)$. Let u and v be the magnitudes of the largest magnitude integers in vectors U and V, respectively. Then we give a $\Sigma_0^{1,b}$ simple tree description of a length m vector W, with structure tree m— $\pm\,(uv+1)$, letting $W_i := (U_i)(V_i)$. Then $U \cdot V := \mathsf{sum}(W)$.

□

A key operation on matrices is of course matrix multiplication. Given an $a \times b$ matrix M and a $(b \times c)$ matrix N, we define the $a \times c$ matrix product $M * N$ in the usual way. To facilitate this definition, we let $\mathsf{col}(N, j)$ be the j^{th} column of N. The length b vector $\mathsf{col}(N, j)$ can be given by a $\Sigma_0^{1,b}$ simple tree definition. Notice that entry (i, j) of matrix M is given by $(M_i)_j$.

Lemma 5.2.2 *$\mathrm{V}^0 + \mathsf{enum}$, $\Delta_1^{1,b}$ defines the* relation *$M * N = W$ and proves that $M * N = W$ and $M * N = W'$ imply $W = W'$.*

Proof

We check that M, N, and W are matrices of legal dimensions. Supposing W is $a \times c$, we check that $\forall i < a \forall j < c\ (W_i)_j = M_i \cdot \mathsf{col}(N, j)$. We can use either set quantifier to refer to $\mathsf{col}(N, j)$.

□

When we write an expression of the form $M * N = W$, we do not mean to assert the existence of $M * N$, though we can assert the existence in a stronger theory.

Lemma 5.2.3 *Matrix multiplication is $\Delta_1^{1,b}$ definable in $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}} - CA$.*

Proof

Suppose M is $a \times b$ and N is $b \times c$. Let m be the absolute value of the largest magnitude number in M or N. Let $r := m^2 b$, a bound on the magnitude of numbers in the matrix product $M * N$. Using the structure tree a—c—r, we give a $\Delta_1^{1,b}$ simple tree definition of $W = M * N$ by $(W_i)_j := M_i \cdot \mathsf{col}(N, j)$.

□

Lemma 5.2.4 *(Associativity of Matrix Multiplication)* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}} - \mathit{CA}$ *proves that for matrices* A*,* B*, and* C*,* $A * (B * C) = (A * B) * C$*.*

Proof

Suppose $L = A * (B * C)$ and $R = (A * B) * C$. To show $L = R$, first we can check that the matrices have the correct dimensions. Then we need to check that for any allowed (i, j), $L(i, j) = R(i, j)$. We can define vectors V_L and V_R, so that $L(i, j) = \mathsf{sum}(V_L)$ and $R(i, j) = \mathsf{sum}(V_R)$. We find these vectors by just following the definition of matrix multiplication and dot product, expanding it all out into one vector, using $\Delta_1^{1,\mathrm{b}}$−CA to assert their existence. They will be vectors of the same length containing the same numbers arranged in two different ways. To show that the two vectors sum to the same number we can define a bijection between the two vectors, equality following by lemma 4.5.6, which uses PHP.

□

Now we want to be able to express that a set of vectors is linearly independent. To do this we will view an $m \times n$ matrix M as a set of m vectors, each of length n, thus the m vectors are $M_{m-1}, \ldots, M_0$. To say that M is linearly independent, we say that the choice $C_{m-1} = \ldots = C_0 = 0$ is the only way to make the linear combination $C_{m-1}M_{m-1} + \ldots + C_0 M_0$ equal the zero vector (or one might say that M has full row rank). Notice that we are discussing such notions for integers, while it is typical to use a field as the elements of the matrices and vectors. Later we will do all the arithmetic modulo a prime p, so that we will be working over the field $\mathbb{F}_p$. However, for now, when not doing modular arithmetic, we have in mind the field of rationals, as will be discussed shortly.

To facilitate the formal description note that given a vector C, $\langle C \rangle$ is in fact a matrix with one row and number of columns equal to the length of C. Thus for an $m \times n$ matrix M and a length m vector C, $\langle C \rangle * M$ is essentially the linear combination $C_{m-1}M_{m-1} + \ldots + C_0 M_0$. We let $\widehat{0}$ be the zero vector, with its length being implicit from context.

Definition 5.2.5 *Let* $\mathit{linIndep}(M)$ *be:*

$$\forall C \ ((\mathit{isVector}(C) \ \wedge \ \mathit{len}(C) = \mathit{len}(M) \ \wedge \ \langle C \rangle * M = \langle \widehat{0} \rangle) \Rightarrow C = \widehat{0}).$$

To say $\mathsf{linIndep}(M)$, means that matrix M is linearly independent. Notice that there is an implicit universal quantifier over the bound on the size of entries in vector C, so we are ultimately ranging over all integer vectors C. Also notice that by $\langle\widehat{0}\rangle$ we mean the $1 \times b$ zero matrix, where M has dimensions $a \times b$.

And now we give a formalization of the dimension principle by formula Dim, saying that if an $m \times n$ matrix is linearly independent then $m \leq n$.

Definition 5.2.6 *Let Dim(M,m,n) be:*

$$\mathit{isMatrix}(M, m \times n) \wedge \mathit{linIndep}(M) \Rightarrow m \leq n.$$

As mentioned, though this principle is stated over the integers, we have in mind the rationals. The above principle is true since the linear independence of a matrix as defined here is equivalent to the usual notion in which the matrix of integers is viewed as a matrix of rationals. The reason for this is that if a matrix is linearly independent over the integers, then it is linearly independent over the rationals, since we can multiply everything though by the denominators to obtain statements involving only integers. The reason we do not work over the rationals is due to the difficulty with summing a sequence of rational numbers (which would be used by the dot product). A similar difficulty was pointed out in the last section following theorem 5.1.12. While summing a sequence of integers is within the scope of enum, for rationals this would require finding common denominators and so essentially contain the power of finding a product of a sequence of natural numbers; doing this explicitly is beyond the scope of bounded arithmetic since the function grows too fast. Products would be allowed under the strict conditions that an appropriately large number existed, conditions we do not have here; products are discussed at the end of section 4.5 in the previous chapter.

Question 5.2.7 *Is there some way for bounded arithmetic to deal with proofs in which one of the steps involves taking the sum of a sequence of rationals?*

We now consider modular versions of these notions. For a dot product we can simply talk about $U \cdot V \pmod p$. For a matrix product, by $M * N = W \pmod p$, we simply mean for the $(\mathsf{mod}\ p)$ operation to be applied to all the entries of the resulting W. We can define linear independence for this context. Let $\mathsf{modLinIndep}(M, p)$ be:

$$\forall C\ ((\mathsf{isVector}(C)\ \wedge\ \mathsf{len}(C) = \mathsf{len}(M)\ \wedge\ \langle C\rangle * M = \langle\widehat{0}\rangle\ (\mathsf{mod}\ p)) \Rightarrow C = \widehat{0}\ (\mathsf{mod}\ p)).$$

Then we can define modular versions of the dimension principle.

Definition 5.2.8

- *Let modDim(p) be:*

$$\forall M, m, n\ (\mathit{isMatrix}(M, m \times n, p) \wedge \mathit{modLinIndep}(M, p) \Rightarrow m \leq n).$$

- *Let* modDim *be:* $\forall p\ (\mathsf{prime}(p) \Rightarrow \mathsf{modDim}(p))$.

Without effecting the discussion, we could impose requirements that vectors and matrices have entries x such that $0 \leq x < p$, but we would still need to do the operations modulo p, so that approach does not appear to buy us anything in terms of presentation.

As mentioned in the beginning of this section we will use these dimension principles by adding them to the theory $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$. It turns out that we can prove the modular versions in V^1, while Dim is not even provable in bounded arithmetic (we discuss this more later). The crucial difference is that we can find multiplicative inverses for the elements of $\mathbb{F}_p$ in V^1 (in fact $\mathrm{I}\Delta_0$ suffices), but not for the integers (that is integers do not have multiplicative integer inverses); we note later on where this difference becomes crucial. The following theorem basically appears in Kaye's book [30] as a theorem in PA, though the proof goes through in $\mathrm{I}\Delta_0$.

Theorem 5.2.9 *[30, theorem 5.2]* $\mathrm{I}\Delta_0$ *proves that for a prime* p, $\forall\ 0 < a < p\ \exists b < p\ ab = 1\ (\mathsf{mod}\ p)$.

We now sketch the informal proof of the dimension principle, a basic linear algebra proof. Suppose $U = \{u_{m-1}, \ldots, u_0\}$ are m linearly independent vectors in some vector space with basis $B = \{b_{n-1}, \ldots, b_0\}$. To show $m \leq n$, we assume for sake of contradiction that $m > n$. We now describe a process:

1. Let $B_0 := B$. Let $i := 0$.
2. If $i = n$ we are done.

   Otherwise consider $B_i = \{v_{n-1}, \ldots, v_0\}$ (it is a basis). Write u_i as a linear combination of vectors from B_i; suppose $u_i = a_{n-1}v_{n-1} + \ldots + a_0 v_0$. For some $j < n$, $v_j \in B$ and $a_j \neq 0$, otherwise U would have a linear dependence.
3. Let $B_{i+1} := B_i - \{v_j\} + \{u_i\}$. We can write v_j as a linear combination of vectors from B_{i+1} *using the fact that we have an inverse to field element* a_j. Thus B_{i+1} is a basis. Set $i := i + 1$ and goto step 2.

The procedure produces the basis $B_n = \{u_{n-1}, \ldots, u_0\} \subseteq U$, thus u_n can be written as a linear combination of vectors from U, contradicting the linear independence of U. The following proof will formalize this process.

Theorem 5.2.10 V^1 *proves* modDim.

Proof

Let p be a prime. Let M be a linearly independent $m \times n$ matrix over $\mathbb{F}_p$, and suppose for contradiction that $m > n$. Using $\Sigma_1^{1,\mathrm{b}}$−IND do induction on r up to n, on the following formula, which says that there is a set of n vectors B that generates the m vectors of M, where the first r vectors of B are in M:

$$\begin{aligned} \exists B \quad & \mathsf{isMatrix}(B, n \times n) \\ \wedge \quad & \forall i < m \; \exists C \; (\mathsf{isVector}(C, n, p) \wedge \langle C \rangle * B = M_i \; (\mathsf{mod}\; p)) \\ \wedge \quad & \forall i < r \; B_i = M_i \end{aligned}$$

Notice that all the sets in the above formula can be bound so it is of complexity $\Sigma_1^{1,b}$. For the base case we can just choose the standard basis. For the inductive step suppose there is a matrix B which works for r; we simulate a step in the previously described process to arrive at matrix B' which works for $r+1$. We have a vector C such that $\langle C \rangle * B = M_r \; (\mathsf{mod}\; p)$. By the linear independence of M there is an i, $r \leq i < n$, such that $C_i \neq 0$ (otherwise, since we have an inverse to 1 in $\mathbb{F}_p$, we would be able to move the M_r to the other side of the equation defining a vector of coefficients on M which violates the linear independence of M). We can find an inverse to C_i by lemma 5.2.9, and thus can move B_i to the right side of the equation, switching the role of B_i and M_r, thus maintaining the generating property. B_i and B_r can then be switched to arrive at B' which works for $r+1$.

□

The lack of appropriate inverses precludes applying the above proof to Dim. In fact, bounded arithmetic cannot prove Dim, an immediate result of the following point (recall EXP from definition 2.1.9).

Lemma 5.2.11 $\mathrm{V}^0 + \mathit{enum}$ *proves* $\mathit{Dim} \Leftrightarrow \mathit{EXP}$.

Proof

EXP easily proves Dim since we can code everything as a number.

For the other direction, suppose we want to show x^n exists, given that we know Dim. We can define M to be the following $(n+1) \times n$ matrix

$$\begin{bmatrix} & & & & 1 \\ & & & 1 & -x \\ & & & -x & \\ & & \cdot & & \\ & & \cdot & & \\ 1 & -x & & & \\ -x & & & & \end{bmatrix}$$

such that $M(i, n-1-i) := 1$ and $M(i+1, n-1-i) := -x$ for $0 \leq i \leq n-1$; the other entries are zero. Since $n+1 > n$, by Dim we know $\neg$linIndep(M), which means that there exists a length $n+1$ non-zero vector $C = \langle C_n, \ldots, C_0 \rangle$ such that $\langle C \rangle * M = \langle \widehat{0} \rangle$. If we write out the corresponding equations we obtain:

$$\begin{aligned} C_1 &= xC_0 \\ C_2 &= xC_1 \\ &\vdots \\ C_n &= xC_{n-1} \end{aligned}$$

We know some $C_i \neq 0$, which propogates through the other C_j, so in fact all the $C_i \neq 0$ (by $\Sigma_0^{1,\mathrm{b}}-$IND). Then $\frac{C_n}{C_0}$ is x^n.

□

Notice that part of the proof of this lemma works for the modular principles obtaining the following (recall modular exponentiation from the end of chapter 4).

Lemma 5.2.12 $\mathrm{V}^0 + enum + modDim$ *defines modular exponentiation for any prime modulus.*

The proof of this lemma is basically the same as the proof of lemma 5.2.11, except that we use the modular principles and all the operations are (mod p) for some prime p. It does not seem that the converse is the case. Namely, it does not appear that having modular exponentiation is enough to code the objects in the proof of modDim. The same proofs could also be modified to define a product of numbers by replacing the n x's by the numbers $x_0, \ldots, x_{n-1}$.

For the reader who finds the dimension principles artificial, at least for modDim she can view the argument as occurring in V^1. Ultimately the naturalness of the dimension principles would have to be shown by a wide context of use, as is the case for the counting principles. Already Soltys and Cook [43] use such principles, and this work provides further use, thus in a sense increasing the evidence for the naturalness of such principles.

5.2.2 The Non-Uniform Fisher Inequality

The formal proof of the Non-Uniform Fisher Inequality follows the proof in [5], though we deviate in presentation due to the non-typical linear algebra principle Dim. By the above point that Dim is equivalent to EXP, this is not surprising (in the presence of EXP we can do pretty much anything, yet we present the proof in a form which could possibly be modified to avoid EXP).

Theorem 5.2.13 $\mathrm{V}^0 + enum + Dim + \Delta_1^{1,\mathrm{b}}-CA$ *proves:*

$$isIncidenceType(H, [n+1], \{\lambda\}, m, n) \Rightarrow m \leq n.$$

Proof

Let H be an appropriate incidence system. As in earlier cases using sunflowers, we can use PHP to cover the case in which one of the sets has only λ many elements, so assume all sets have at least $\lambda+1$ elements. We now show $\mathsf{linIndep}(H, m \times n)$, which by Dim yields the result of $m \leq n$. Let C be any length m vector of unary integers, and suppose $\langle C \rangle * H = \langle \widehat{0} \rangle$; our goal is to show $C = \widehat{0}$. The matrix transpose H^t is easily defined (recall the transpose is defined by converting an $m \times n$ matrix into an $n \times m$ matrix defining entry (i, j) in the transpose to be be (j, i) in the original). So using associativity we have:

$$\langle C \rangle * (H * H^t) = (\langle C \rangle * H) * H^t = \widehat{0} * H^t = \widehat{0}.$$

Notice that $M := H * H^t$ is an $m \times m$ matrix in which entry (i, j) is $\|H_i \cap H_j\|$, so for $i \neq j < m$, $M(i, j) = \lambda$, and $M(i, i) = \lambda + a_i$, for some $a_i > 0$. Thus $\langle C \rangle * (H * H^t) = \widehat{0}$ implies that we have the following system of equations:

$$\begin{array}{ll} (\text{sum } 0) & C_{m-1}\lambda + \ldots + C_0\lambda + C_0 a_0 = 0 \\ \vdots & \vdots \\ (\text{sum } m-1) & C_{m-1}\lambda + \ldots + C_0\lambda + C_{m-1} a_{m-1} = 0 \end{array}$$

We have just written down rearrangements of the m dot products. The calculation $C_i * M(i, i) = C_i * (\lambda + a_i) = C_i\lambda + C_i a_i$ is one step of the rearrangement. Then we move the $C_i a_i$ part to one end of the sum (this movement keeps the sum the same by lemma 4.5.6 since we can define a bijection between these 2 ways of summing). Now assume for contradiction that some $C_i \neq 0$. Let $r := C_{m-1}\lambda + \ldots + C_0\lambda$, be the sum of the first m terms in each of the above sums. If $C_i > 0$, since $a_i > 0$, by sum i, we would have to have $r < 0$. Since all the sums have the same r, each $C_j > 0$ by sum j, for $0 \leq j < m$. But then the sums in the above system cannot equal zero. The case for $C_i < 0$ is similar. So in fact we must have that $C = \widehat{0}$, finishing the proof.

□

Question 5.2.14 *Can the Non-Uniform Fisher Inequality (the statement in the theorem 5.2.13) be proved in bounded arithmetic?*

5.2.3 The RCW theorem

We now formalize the RCW theorem. It uses polynomials, so we will begin with a diversion into polynomials. Though we only use them for this theorem, they could have wide application beyond just this use.

Polynomials

We will first be concerned with defining the polynomials essentially as certain kinds of sequences, and then we will define an evaluation operation on polynomials. Our polynomials will be defined over the field $\mathbb{F}_p$ (for some prime p) and use some set of variables $\{x_{n-1}, \ldots, x_0\}$. Though we could define the polynomial objects over the field of rationals, we would have problems with evaluation since we do not know how to sum a sequence of rationals (this difficulty is discussed after definition 5.2.6); since we can avoid the use of such polynomials we will not worry about them. By a **monomial** we mean a "product expression" of the form:

$$x_{n-1}{}^{d_{n-1}} \ldots x_1{}^{d_1} x_0{}^{d_0},$$

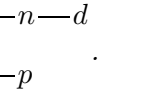
where the $d_i \geq 0$ are integers; if all the $d_i = 0$, we consider this expression to be simply 1. We use the term "expression" because we are not really taking products and sums of numbers, but forming expressions with the symbols $x_{n-1}, \ldots, x_0$; when we evaluate a polynomial with some variable assignment we will actually take products and sums of numbers. By a **monomial term** we mean an expression of the form $c x_{n-1}{}^{d_{n-1}} \ldots x_0^{d_0}$, where $c \in \mathbb{F}_p$ is called a **coefficient**; we call some $x_i{}^{d_i}$ a **term** in the corresponding monomial. By a **polynomial** we mean a sum of monomial terms; note that monomials may be repeated and monomial terms may have zero coefficients. The **degree of a variable** in a monomial is the exponent of that variable. The **degree of a monomial** (or monomial term) is the sum of the degrees of its variables (i.e. $d_{n-1} + \ldots + d_0$), and the **degree of a polynomial** is the degree of a monomial with maximum degree.

Now we discuss our coding of polynomials by simple trees. A polynomial over $\mathbb{F}_p$ will essentially be a pair of sequences, a **coefficient sequence** and a **monomial sequence**. The coefficient sequence is just a vector of length m with elements in $[0, p)$, which we write slightly informally as $\langle C_{m-1}, \ldots, C_0 \rangle$. A corresponding monomial sequence is a sequence of m monomials $\langle M_{m-1}, \ldots, M_0 \rangle$, where each monomial M_i is just a length n vector with elements in $[0, d)$ for some number d which will be a bound on the degree of a variable; so $(M_i)_j$, for $j < n$, tells us what the degree of x_j is in the i^{th} monomial. Given two such sequences, the associated polynomial is:

$$C_{m-1} M_{m-1} + \ldots + C_0 M_0.$$

Let $\mathsf{monomialSeq}(M, m, n, d)$ be: $\mathsf{isSimpleTree}(m — n — d, M)$. Now we can define what we mean by a polynomial (we use $\mathbb{F}_p$ as a suggestive notation for indicating how the parameter p is to be used).

Definition 5.2.15 *Let* $\mathsf{isPoly}(Q, \mathbb{F}_p, m, n, d)$ *be:* $\mathsf{isSimpleTree}(t, Q)$*, where* t *is the structure tree:*



.

Given a polynomial Q, by $\mathbf{deg}(Q)$ we mean its degree.

Lemma 5.2.16 deg *is a* $\Delta_1^{1,b}$ *definable function in* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$.

Proof

Suppose our input is Q such that $\mathsf{isPoly}(Q, \mathbb{F}_p, m, n, d)$. We define D such that $\mathsf{isVector}(D, m, nd)$ by $D_i := \mathsf{sum}((Q_0)_i)$, which is $\Delta_1^{1,b}$ because of the sum function. D_i, for $i < m$, gives the degree of monomial i, a number $\leq n(d-1) < nd$. **deg** is now just the maximum unary integer in vector D.

□

To evaluate a polynomial Q (over field $\mathbb{F}_p$ and n variables), we will be given a variable assignment A, which is just a length n vector with entries in $\mathbb{F}_p$. $\mathbf{eval}(Q, A)$ is the element of $\mathbb{F}_p$ that results from evaluating Q with variable x_i assigned to A_i. To define **eval** we will carry out the evaluation *without working modulo* p, only applying the $\mathsf{mod}\ p$ operation at a few select points. Thus, though the end result is only a number $< p$, the evaluation will involve intermediate values of size about p^b, where b is the degree of polynomial Q. So the evaluation will require that we know p^b exists, thus **eval** will have p^b as an *implicit argument*. This will be fine for our applications where we typically work with low degree polynomials. It is of course tempting to look for a definition of evaluation that does not use the existence of p^b. It seems that such a definition would rely on answering the open question discussed after lemma 4.5.7, as to whether or not modular exponentiation can be defined in $\mathrm{I}\Delta_0$. We will give a formal definition of evaluation for monomials calling this function eval. Once we have that, we define evaluation for polynomials in terms of it, using the same notation eval (though there will never be confusion about what function we have in mind). For a a polynomial P (with m monomials) and assignment A we define:

$$\mathsf{eval}(P, A) := \sum_{i<m} (P_1)_i \mathsf{eval}((P_0)_i, A) \ (\mathsf{mod}\ p).$$

This definition really involves defining a vector and then summing it, though we write it in this form to facilitate the readability of a number of arguments.

Lemma 5.2.17 eval *(for monomials) is a* $\Delta_1^{1,b}$ *definable function in* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$.

Proof

Suppose the input consists of a monomial M, such that $\mathsf{isVector}(M, n, d)$, and an assignment vector A such that $\mathsf{isVector}(A, n, p)$. Suppose $b := \mathsf{deg}(M)$ and p^b is an implicit argument; checking the degree uses the theory indicated in the lemma, $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$. We will define $\mathsf{eval}(M, A; p^b)$. We will first evaluate M at each of its terms,

by defining a vector E, such that $\mathsf{isVector}(E, n, p)$. For $i < n$, let $E_i := A_i^{M_i} \ (\mathsf{mod}\ p)$. Taking the power is justified by the fact that $M_i \leq b$ (so the power is no bigger than p^b before applying the $(\mathsf{mod}\ p)$ operation), which we obtain by an easy application of PHP, injecting the 1's of M_i into the 1's of M.

Now we would like to take the product of E, but it is of length n, so if $b < n$, p^n may not exist. However, we have at most b non-one terms, so our approach will be to define a vector R which will be E with it's 1's removed; we can then take the product of R. First we define E' from E, by making any non-one unary integer of E into a single 1 and zeroing out the rest. Let F enumerate the c elements of E', that is F enumerates where the non 1 entries of E appear. Notice that $c \leq \mathsf{min}(n, b) \leq b$, so p^c exists. Now we can define the vector R such that $\mathsf{isVector}(R, c, p)$; for $j < c$ we let $R_j :=$ the entry of E that begins at bit $F(j)$. Now we can define $\mathbf{eval}(M, A)$ to be $\mathsf{product}(R, c, p)$, since p^c exists. Recall that product is done $(\mathsf{mod}\ p)$.

□

Note that *within* our definition of eval (for monomials), operations are *not* always done $(\mathsf{mod}\ p)$. However now that we have this definition we will not be explicitly talking about the inner workings of evaluation, thus whenever we use eval in arithmetic operations we are always working $(\mathsf{mod}\ p)$. From here on out we use the following convention: *All operations involving eval are performed* $(\mathsf{mod}\ p)$*, though this will not be mentioned explicitly.*

A key point about our definition of evaluation will be that it will respect various operations. A simple fact of this sort is the following fact.

Lemma 5.2.18 $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}} - \mathsf{CA}$ *proves that for a monomial* M *such that* $\mathsf{isVector}(M, n, d)$*, a number* $x < d$*, a number* $a < p$*, and an assignment* A *such that* $\mathsf{isVector}(A, n, p)$*:*

$$\mathsf{eval}(\langle x \rangle \frown M, \langle a \rangle \frown A) = a^x \mathsf{eval}(M, A).$$

For other facts showing that evaluation respects operations (like adding and multiplying them) we will want to work with sequences of polynomials. For a sequence of polynomials we will want them to be over the same variables and field $\mathbb{F}_p$. But furthermore, we will in fact only need two kinds of such sequences for our work: Sequences in which each polynomial in the sequence is of the same type (a **simple polynomial sequence**), and sequences of length 2 in which the polynomials can be of different types (though the variables and field are the same).

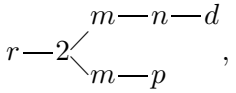
Definition 5.2.19 *Let* $\mathsf{isSimplePolySeq}(S, r, \mathbb{F}_p, m, n, d)$ *be:* $\mathsf{isSimpleTree}(r \text{—} \mathsf{t}, S)$*, where* t *is the structure tree from the definition of* isPoly*.*

Notice that the simple polynomial sequence is a string tree in which each of the immediate r children to the root are polynomials of type t. If we just refer to a **polynomial sequence** we

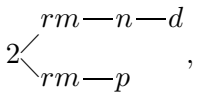
mean a simple polynomial sequence or a pair of polynomials (over the same variables and field). We do not deal with more complicated polynomial sequences because some aspects use a stronger theory, their presentation is more cumbersome, and we have no need to use them. From here on out, "polynomials" and "polynomial sequences" will always refer to our codings. We will often just refer to some, but not all, of the parameters, being colloquial in referring to the parameters (e.g. the polynomial having n variables or m monomials, or being over $\mathbb{F}_p$). Supposing a polynomial or polynomial sequence is defined over $\mathbb{F}_p$, with n variables, we say an assignment A is **appropriate** for it if $\mathsf{isVector}(A, n, p)$. When we refer to polynomials and/or polynomial sequences together, we generally expect them to be over the same variables and field, in which case we say that they **correspond**.

We will define a function **polySum** which maps a sequence of polynomials to a single polynomial, its sum, without collecting together monomial terms with the same monomial. Precisely, for a polynomial sequence S, **polySum**(S) is the polynomial Q whose monomial sequence, Q_0, is the concatenation $(S_{r-1})_0 \frown \ldots \frown (S_1)_0 \frown (S_0)_0$, and whose coefficient sequence, Q_1, is the concatenation

$(S_{r-1})_1 \frown \ldots \frown (S_1)_0 \frown (S_0)_1$. For a simple polynomial sequence S with structure tree:



polySum(Q) will have the following structure tree :



and in this case it can be defined by a $\Sigma_0^{1,b}$ simple tree description. We can give an analogous $\Sigma_0^{1,b}$ simple tree description for a polynomial sum of two polynomials, not necessarily with the same structure. Thus we have proved the following lemma.

Lemma 5.2.20 polySum *is* $\Sigma_0^{1,b}$ *bit definable.*

Now we note that extending a polynomial sequence with a single polynomial works right with respect to polynomial summation. The following lemma which we write more intuitively as:

$$\mathsf{polySum}(\langle Q\rangle \frown S) = Q + \mathsf{polySum}(S),$$

is a consequence of polynomial summation defined by concatenation.

Lemma 5.2.21 *For S a polynomial sequence and a corresponding polynomial Q, V^0 proves that*

$$\mathit{polySum}(\langle Q\rangle \frown S) = \mathit{polySum}(\langle Q\rangle \frown \langle \mathit{polySum}(S)\rangle).$$

In the work on polynomials, it will improve readability to use the abuse of notation preceding the lemma to refer to adding (or multiplying by) a single polynomial. Notice that polynomial summation is not commutative as defined. The order of the terms matters, which will be important to keep in mind especially for polynomial products where we multiply the polynomials out in a particular way.

A key property is to connect polynomial summation with number summation via the evaluation function, showing that evaluation respects summation.

Lemma 5.2.22 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$ *proves that for corresponding polynomials* P *and* Q*, and an appropriate assignment* A*,*

$$\mathit{eval}(P, A) + \mathit{eval}(Q, A) = \mathit{eval}(P + Q, A).$$

Proof

Since polynomial summation $P + Q$ just concatenates the monomial lists, the right hand side is a sum which the left hand side just breaks into two pieces (allowed by lemma 4.5.4).

□

Now we want to extend this to a simple sequence of polynomials $P_{r-1}, \ldots, P_0$. We want to show that for any appropriate variable assignment A we have

$$\mathsf{eval}(P_{r-1} + \ldots + P_0, A) = \mathsf{eval}(P_{r-1}, A) + \ldots + \mathsf{eval}(P_0, A).$$

We will work in $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ where we can freely use the eval function in the comprehension axioms. Given a polynomial sequence S of length m and an appropriate assignment A, let $S^{(A)}$ be the length m vector defined (for $i < m$) by

$$(S^{(A)})_i := \mathsf{eval}(S_i, A).$$

This operation is valid in $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ (the existence of the vector uses $\Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$). Recall the definition of "$T_{<k}$" immediately following lemma 4.4.21.

Lemma 5.2.23 *(Sum respects Evaluation) We work in* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$*. Suppose that* S *is a simple polynomial sequence,* $Q = \mathit{polySum}(S)$*, and* A *is an appropriate assignment. Then we can prove*

$$\mathit{eval}(Q, A) = \mathit{sum}(S^{(A)}).$$

Proof

We now have a fixed S, $S^{(A)}$, and A, which will be parameters in our induction. Suppose S is of length r. It then suffices to show by $\Delta_1^{1,b}$ induction on k up to r that:

$$\mathsf{eval}(\mathsf{polySum}(S_{<k}), A) = \mathsf{sum}((S^{(A)})_{<k}).$$

For the inductive step, note that:

$$\begin{aligned} \mathsf{eval}(\mathsf{polySum}(S_{<k+1}), A) &= \mathsf{eval}(S_k + \mathsf{polySum}(S_{<k}), A) \\ &= (S^{(A)})_k + \mathsf{sum}((S^{(A)})_{<k}) \\ &= \mathsf{sum}((S^{(A)})_{<k+1}). \end{aligned}$$

The first equality uses lemma 5.2.21. The second equality uses the inductive hypothesis and lemma 5.2.22. The last equality follows from lemma 4.5.5.

□

We will now define a product of polynomials. Suppose we have a polynomial sequence $S = \langle S_{r-1}, \ldots, S_0 \rangle$, where polynomial S_i is $M_{m-1}^{(i)} + \ldots + M_0^{(i)}$, each $M_j^{(i)}$ being a monomial term. So the product of S should be the following:

$$(M_{m-1}^{(r-1)} + \ldots + M_0^{(r-1)}) \ldots (M_{m-1}^{(0)} + \ldots + M_0^{(0)}),$$

We will multiply this product out into a single polynomial Q which looks like:

$$M_{m-1}^{(r-1)} \ldots M_{m-1}^{(1)} M_{m-1}^{(0)} + \ldots + M_0^{(r-1)} \ldots M_0^{(1)} M_1^{(0)} + M_0^{(r-1)} \ldots M_0^{(1)} M_0^{(0)}.$$

Note that we have a particular lexicographical order in mind, which will be important for showing that products respect evaluation. We denote this operation by $\mathbf{polyProd}(S) = Q$.

Notice that the number of monomials in Q is m^r, so for products we will assume that m^r exists. The intuition is that the number of polynomials in the sequence (i.e. r) must be small, though the length of each such polynomial (i.e. m) need not be too small. Since Q also involves products of length r from $\mathbb{F}_p$ (for the coefficients) we will require that p^r exists too; m^r and p^r will both be implicit arguments to **polyProd**.

Lemma 5.2.24 polyProd *is a* $\Sigma_0^{1,b}$ *bit definable function.*

Proof

Consider the case of S a simple polynomial sequence; for a pair of polynomials it is analogous. Suppose S has structure $\mathsf{t} := r \text{—} 2 \begin{array}{l} \diagup\, m \text{—} n \text{—} d \\ \diagdown\, m \text{—} p \end{array}$. We define $\mathsf{polyProd}(S) = Q$

by giving a $\Sigma_0^{1,b}$ simple tree description of Q with structure tree $\mathsf{s} := 2\begin{array}{l}\diagup\, m^r\text{—}n\text{—}dr \\ \diagdown\, m^r\text{—}p\end{array}$.

Suppose $a = (a_0, a_1, a_2) \trianglelefteq s$, where $a_1 < m^r$, viewed as a base m number, is taken to be a sequence $c = (c_{r-1}, \ldots, c_0)$ where each $c_i < m$; a_2 will not be present if $a_0 = 1$. We now define the value of Q_a.

If $a_0 = 1$, then we need to define the coefficient of the monomial term corresponding to a, which is: $\prod_{i<r}(S_{\langle i,1,c_i\rangle})$. For the latter product we give a simple tree description of the sequence and apply $\mathsf{product}$ to it with the implicit size argument of p^r; this can be done in V^0.

If $a_0 = 0$, then $a_2 < n$ and we need to define the exponent of variable x_{a_2}, for the monomial term corresponding to a_1, which is: $\sum_{i<r}(S_{(i,0,c_i,a_2)})$, since it is a short summation (p^r exists) we can get by without enum.

□

Now we show that the polynomial product definition makes sense when we extend a polynomial sequence by a single polynomial.

Lemma 5.2.25 V^0 *proves that for polynomial sequence S and a corresponding polynomial Q*

$$\mathit{polyProd}(\langle Q\rangle \frown S) = Q * \mathit{polyProd}(S).$$

Proof

Consider the case of S such that $\mathsf{isSimplePolySeq}(S, r, \mathbb{F}_p, m, n, d)$ and Q such that $\mathsf{isPoly}(Q, \mathbb{F}_p, m, n, d)$. Let $A := \mathsf{polyProd}(\langle Q\rangle \frown S)$, and $B := Q{*}\mathsf{polyProd}(S)$. $\langle Q\rangle \frown S$ is a simple tree of type $r+1$—$2\begin{array}{l}\diagup\, m\text{—}n\text{—}d \\ \diagdown\, m\text{—}p\end{array}$. A is a polynomial of type

$$\mathsf{t} := 2\begin{array}{l}\diagup\, m^{r+1}\text{—}n\text{—}(r+1)d \\ \diagdown\, m^{r+1}\text{—}p\end{array},$$

and B is a polynomial of the same type, though better thought of as:

$$2\begin{array}{l}\diagup\, mm^r\text{—}n\text{—}(d+dr) \\ \diagdown\, mm^r\text{—}p\end{array}.$$

Now we just note that for any $a = \langle a_0, a_1, a_2\rangle \trianglelefteq \mathsf{t}$, $A_a = B_a$. The key fact is that for $a_1 < m^{r+1} = mm^r$, viewed as $\langle c_r, c_{r-1}, \ldots, c_0\rangle$, $c_i < m$, or $\langle e_r, \langle e_{r-1}, \ldots, e_0\rangle\rangle$, $e_i < m$, we get the same sequence (i.e. $c_i = e_i$ for $i < r+1$). Thus the sequence product used for the coefficients, and the sum used for the exponent sum are the same (use lemma 4.5.9 for the product and a similar lemma for the sum; note that we do not need the full blown lemma 4.5.5 for the sum).

□

A helpful lemma for showing that evaluation works right is a general lemma about distributivity, for which we make the following definition.

Definition 5.2.26 *For vectors S and T of lengths n and k respectively, by $S * T$ we mean the vector of length nk defined by (for $j < n$, and $i < k$)*

$$(S * T)_{\langle j,i\rangle} := S_j * T_i.$$

Lemma 5.2.27 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}} - \mathit{CA}$ *proves that for vectors S and T,*

$$\mathit{sum}(S) * \mathit{sum}(T) = \mathit{sum}(S * T).$$

Proof

We can proceed by induction on k on the following $\Delta_1^{1,b}$ formula:

$$\mathsf{sum}(S_{<k}) * \mathsf{sum}(T) = \mathsf{sum}(S_{<k} * T).$$

This reduces to just showing that for S being a single number the claim holds. For S being a single number, we can then do a similar induction on T, reducing the entire problem to basic arithmetic properties.

□

In showing that polynomial products respect evaluation, a key step in the induction uses the following special case for the product of 2 polynomials (similar to lemma 5.2.22 for sums).

Lemma 5.2.28 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}} - \mathit{CA}$ *proves that for corresponding polynomials P and Q, and an appropriate assignment A:*

$$\mathit{eval}(P, A) * \mathit{eval}(Q, A) = \mathit{eval}(P * Q, A).$$

Proof

Suppose $\mathsf{isPoly}(P, \mathbb{F}_p, m, n, d)$ and $\mathsf{isPoly}(Q, \mathbb{F}_p, k, n, e)$. We do a calculation, where over a double sum the first index (in our case, j) is more significant.

$$\begin{aligned}
\mathsf{eval}(P, A) * \mathsf{eval}(Q, A) &= \left(\sum_{j<m} (P_1)_j \mathsf{eval}((P_0)_j, A)\right)\left(\sum_{i<k} (Q_1)_i \mathsf{eval}((Q_0)_i, A)\right) && (5.1)\\
&= \sum_{j<m, i<k} ((P_1)_j \mathsf{eval}((P_0)_j, A))((Q_1)_i \mathsf{eval}((Q_0)_i, A)) && (5.2)\\
&= \sum_{j<m, i<k} P_{\langle 1,j\rangle} Q_{\langle 1,i\rangle} \mathsf{eval}((P_0)_j, A)\mathsf{eval}((Q_0)_i, A) && (5.3)\\
&= \sum_{j<m, i<k} (P * Q)_{\langle 1,(j,i)\rangle} \mathsf{eval}((P * Q)_{\langle 0,\langle j,i\rangle\rangle}, A) && (5.4)\\
&= \mathsf{eval}(P * Q, A) && (5.5)
\end{aligned}$$

Line 5.2 works because of lemma 5.2.27. For line 5.4 note that $(P * Q)_{\langle 0,\langle j,i\rangle\rangle} = P_{\langle 0,j\rangle} * Q_{\langle 0,i\rangle}$, so proving the following claim suffices (note that this claim is basically the lemma, but for monomials).

Claim 5.2.29 *For monomials M and N such that $\mathsf{isVector}(M, n, d)$ and $\mathsf{isVector}(N, n, e)$, we have:*

$$\mathsf{eval}(M, A) * \mathsf{eval}(N, A) = \mathsf{eval}(M * N, A).$$

To prove the claim, we show by induction on r up to n that:

$$\mathsf{eval}((M * N)_{<r}, A) = \mathsf{eval}(M_{<r}, A) * \mathsf{eval}(N_{<r}, A).$$

For the inductive step note the calculation:

$$\begin{aligned}
\mathsf{eval}((M * N)_{<r+1}, A) &= A_r^{(M*N)_r} \mathsf{eval}((M * N)_{<r}, A)\\
&= A_r^{M_r} A_r^{N_r} \mathsf{eval}(M_{<r}, A)\mathsf{eval}(N_{<r}, A)\\
&= \mathsf{eval}(M_{<r+1}, A)\mathsf{eval}(N_{<r+1}, A)
\end{aligned}$$

The second equality uses the inductive hypothesis. The first and last equality, use lemma 5.2.18.

□

Now in showing that evaluation respects products, note that we have a significant added restriction on the size of the parameters unlike in sums, requiring that both p^r and m^r exist. The

proof will follow the natural inductive proof. To show that a product of r polynomials evaluates to the same value as the multiplied out polynomial, we essentially proceed by induction on r. The formal proof will proceed by induction up to r on the one fixed polynomial sequence, depending on the fact that the multiplied out polynomial is done in a certain order. The following lemma has a proof similar in form to that of lemma 5.2.23, though it uses the properties of products.

Lemma 5.2.30 *(Product respects Evaluation) We work in* $\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$. *Suppose that* S *is a polynomial sequence of length* r, $Q = \mathsf{polyProd}(S)$, *and* A *is an appropriate assignment, all over the field* $\mathbb{F}_p$. *Suppose* p^r *and* p^b *exist, where* b *is the degree of* Q *(the degree of the* S_i *are smaller). Then we can prove*

$$\mathsf{eval}(Q, A) = \mathsf{product}(S^{(A)}).$$

Proof

We now have a fixed S, $S^{(A)}$, and A, which will be parameters in our induction. It then suffices to show by $\Delta_1^{1,\mathrm{b}}\text{-}\mathsf{IND}$ on k upto r that:

$$\mathsf{eval}(\mathsf{polyProd}(S_{<k}), A) = \mathsf{product}(S^{(A)}_{<k}).$$

For the inductive step, note that:

$$\begin{aligned} \mathsf{eval}(\mathsf{polyProd}(S_{<k+1}), A) &= \mathsf{eval}(S_k * \mathsf{polyProd}(S_{<k}), A) \\ &= (S^{(A)})_k * \mathsf{product}((S^{(A)})_{<k}) \\ &= \mathsf{product}((S^{(A)})_{<k+1}) \end{aligned}$$

The first equality follows from lemma 5.2.25. The second equality uses the inductive hypothesis and lemma 5.2.28. The last follows from lemma 4.5.9.

□

We can talk about polynomials that look different as being equal if they have the same values for all variable assignments.

Definition 5.2.31 *For polynomials* P *and* Q, *both over* $\mathbb{F}_p$ *and* n *variables, by* $(P \equiv Q)$ *we mean:*

$$\forall A \ (\mathsf{isVector}(A, n, p) \Rightarrow \mathsf{eval}(P, A) = \mathsf{eval}(Q, A).$$

In order to obtain the desired bounds we will be interested in a certain kind of polynomial called **multi-linear polynomials**, which means that the largest degree of any variable is 1. To refer to such polynomials we simply let $\mathsf{isMLin}(Q, \mathbb{F}_p, m, n)$ state that $\mathsf{isPoly}(Q, \mathbb{F}_p, m, n, 1)$. It often suffices to just evaluate a polynomial at the field elements 0 and 1. In such a case $x_i^2 = x_i$, so we

can replace x_i to any power with simply x_i as far as such evaluation is concerned. This process yields a multi-linear polynomial and so we call it **multi-linearizing**. Given a polynomial P, we let $\mathbf{MLize}(P)$ be the result of applying this process to P (i.e. replacing x_i^e by x_i); note that no collection or rearrangement of the monomial terms occurs. Simply replacing nonzero exponents by 1 is easy, so we can $\Sigma_0^{1,b}$ bit define **MLize**. Note that the multi-linearized polynomial preserves evaluation for 0/1 values, and it is easy to show this (though we require the stronger theory in order to have eval as a function).

Lemma 5.2.32 *We work in* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$. *Suppose* P *is a polynomial over* $\mathbb{F}_p$ *and* n *variables. Suppose that* $\mathsf{isVector}(A, n, 2)$. *Then* $\mathsf{eval}(P, A) = \mathsf{eval}(\mathsf{MLize}(P), A)$.

The goal of multi-linearizing is to arrive at a polynomial with fewer monomial terms. However as defined we have the same number of monomial terms. The savings will come by consolidating monomial terms that have the same monomial. For example, the polynomial $2x_1x_3+x_2x_3+3x_1x_3+5x_2x_3$ could be consolidated into the polynomial $5x_1x_3 + 6x_2x_3$.

We will now discuss how to consolidate monomial terms in general, though we will only use it in application to a multi-linearized polynomial. Suppose we have a polynomial P and we want to consolidate its like monomial terms. One immediate difficulty is that two different polynomials with the same structure tree may become polynomials with very different structure trees when all the monomial terms are consolidated; it's unclear how to account for this with simple trees. To get around this we will define the simple tree structure of the consolidated polynomial in advance, using a monomial sequence. Suppose we have a polynomial P such that $\mathsf{isPoly}(P, \mathbb{F}_p, m, n, d)$, and a monomial sequence M such that $\mathsf{monSeq}(M, k, n, e)$ with no repeated monomials, so they have the same number of variables, but otherwise can be different (typically $k < m$). Also, suppose that every monomial appearing in P appears in M. We define $\mathbf{consolidate}(P, M)$ to be the unique polynomial with monomial sequence M that is equal (under evaluation) to P. To define $\mathbf{consolidate}(P, M)$, we define its coefficients by freely using the sum function on the appropriate coefficients of P, so we obtain the following lemma.

Lemma 5.2.33 consolidate *is* $\Delta_1^{1,b}$ *definable in* $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$.

Proof

Suppose $\mathsf{isPoly}(P, \mathbb{F}_p, m, n, d)$ and $\mathsf{isMonSeq}(M, k, n, e)$ and $\forall i < m\ \exists j < k\ (P_0)_i = M_j$ and $\forall i < j < k\ M_i \neq M_j$. We define $Q := \mathsf{consolidate}(P, M)$ such that $\mathsf{isPoly}(Q, \mathbb{F}_p, k, n, e)$. $Q_0 := M$. The main part is to define $(Q_1)_i$ for $i < k$. For each i, we consider monomial M_i and let F enumerate all the b positions $j < m$ such that $M_i = (P_0)_j$. Then define $(Q_1)_i := \sum_{j<b}(P_1)_{F_i(j)}$, the sum of the appropriate coefficients from polynomial P.
□

And of course we want to know that we can prove a polynomial is equal to one of its consolidated forms. Given that $Q := \mathsf{consolidate}(P, M)$, we want to show that $P \equiv Q$. To evaluate Q we basically evaluate P under the rearrangement given by M. Since we can describe a bijection between these two sums, they will be equal by lemma 4.5.6.

Lemma 5.2.34 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}} - \mathit{CA}$ *proves that if P is a polynomial and M is an appropriate monomial sequence, then $P \equiv \mathsf{consolidate}(P, M)$.*

Proof

Let P, M, and Q be as in the previous lemma. Let A be such that $\mathsf{isVector}(A, n, p)$. Now we show that $\mathsf{eval}(P, A) = \mathsf{eval}(Q, A)$.

$$\begin{aligned} \mathsf{eval}(Q, A) &= \sum_{i<k} (Q_1)_i \mathsf{eval}((Q_0)_i, A) \\ &= \sum_{i<k} (Q_1)_i \mathsf{eval}(M_i, A) \end{aligned}$$

The single coefficient $(Q_1)_i$ is a sum of coefficients from P_1 (say $(Q_1)_i = (P_1)_{b_1} + \ldots + (P_1)_{b_i}$). We essentially want to expand out the last sum to express this. We will define a vector R so that the last sum equals $\mathsf{sum}(R)$. Let b be the maximum b_i, for $i < k$. This can be defined by defining vector B such that $B_i :=$ number of monomials in P_0 that equal M_i; we then set b equal to the maximum number in B. We can now define R to be a vector of length kb, where R_x for $x = \langle x_1, x_0 \rangle$, $x_1 < k, x_0 < b$, is defined to be coefficient x_0 in $(Q_1)_{x_1}$; it is zero if there is no such coefficient. Now consider

$$\mathsf{eval}(P, A) = \sum_{j<m} (P_1)_j \mathsf{eval}((P_0)_j, A).$$

By lemma 4.5.6, it suffices to define a bijection I from the above sum (a sum on a length m vector) to R (a length kb vector). For I, given $j < m$, find $i < k$ such that $(P_0)_j = M_i$, and let F (as in previous lemma) count occurrences of M_i in P_0. Suppose $(P_0)_j$ is occurrence $y < b$. I maps to $\langle i, y \rangle$ in R.

□

We now tie up the work on polynomials with linear algebra. The set of polynomials over $\mathbb{F}_p$ with n variables, with degree $\leq d$, is a vector space with the usual polynomial addition and multiplication by scalars from $\mathbb{F}_p$. A basis is the set of all monomials with degree $\leq d$. We will actually be concerned with a similar vector space: The same set of polynomials, except that we only consider multilinear polynomials. Again, the set of all multilinear polynomials with degree $\leq d$ is a basis. Thus this multi-linear polynomial space has dimension $\binom{n}{d} + \binom{n}{d-1} + \ldots + \binom{n}{0}$.

We want to be able to express that a set of polynomials is linearly independent and then connect this to the notion we have already discussed, the linear independence of a matrix. We now introduce some terminology to discuss this.

The **zero polynomial** on n variables is the polynomial consisting of the single monomial, the constant 0; we call this the $\mathsf{zeroPoly}$. We can clearly $\Sigma_0^{1,b}$ define such a polynomial and prove in V^0 that it evaluates to zero for all variable assignments.

We will want to state that a set of polynomials is linearly independent. To facilitate this we define a dot product operation on polynomials. Suppose $S := \langle S_{m-1}, \ldots, S_0 \rangle$ is a polynomial sequence and $C := \langle C_{m-1}, \ldots, C_0 \rangle$ is a vector, both of length m and over $\mathbb{F}_p$. Let $C \cdot S$ be the polynomial $C_{m-1} * S_{m-1} + \ldots + C_0 * S_0$; whether we mean vector or polynomial dot product will be clear from context. Technically we define a polynomial sequence S' to be the polynomial sequence S in which $(S')_i$ has its coefficients multiplied by C_i. We then do $\mathsf{sumPoly}(S')$. $C \cdot S$ is defined by a $\Sigma_0^{1,b}$ simple tree definition in V^0, so has a $\Sigma_0^{1,b}$ bit definition.

Definition 5.2.35 *Let* $\mathit{linIndepPoly}(S)$ *be:*

$$\mathit{isSimplePolySeq}(S, r, \mathbb{F}_p, m, n, d) \wedge \forall C \ ((\mathit{isVector}(C, n, p) \wedge \ C \cdot S \equiv \mathit{zeroPoly}) \Rightarrow C = \widehat{0} \ (\mathit{mod}\ p)).$$

We could now formulate a principle similar to the dimension principle, except for polynomials. However we can essentially show how to reduce such a principle to the dimension principle already given. Given a sequence of polynomials S, we say it is **canonical** if there is a single monomial sequence $\langle M_{m-1}, \ldots, M_0 \rangle$ such that every polynomial in S is expressed as a linear combination $c_{m-1}M_{m-1} + \ldots + c_0 M_0$.

Definition 5.2.36 *Let* $\mathit{isCanonical}(S)$ *be:* $\exists M \ \forall i < \mathit{len}(S) \ (S_i)_0 = M$.

In this case we can work with a sequence of coefficients $(c_{m-1}, \ldots, c_0)$ as if it is a polynomial by keeping the fixed, ordered sequence of monomials in the background. We can consider S to be a sequence of coefficient sequences, in other words a matrix. In general, if $\mathsf{isSimplePolySeq}(S, r, \mathbb{F}_p, m, n, d)$ we can associate a **coefficient matrix** to S, called $\mathbf{CMatrix}(S)$. It is the $r \times m$ matrix in which row i is the length m coefficient sequence of polynomial S_i. **CMatrix** is $\Sigma_0^{1,b}$ bit definable. If a canonical polynomial sequence S happens to be linearly independent then so will this matrix. The basic intuition is that for a canonical polynomial sequence, evaluation reduces to matrix operations.

Lemma 5.2.37 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}} - \mathit{CA}$ *proves:*

$$\mathit{isSimplePolySeq}(S) \wedge \mathit{isCanonical}(S) \wedge \mathit{linIndepPoly}(S) \Rightarrow \mathit{linIndep}(\mathit{CMatrix}(S)).$$

Proof

Suppose S is of length r with each S_i having m monomials. Let M be the monomial sequence associated with S. Let $B := \mathsf{CMatrix}(S)$, an $r \times m$ matrix. Suppose C is such that $\mathsf{isVector}(C, r, p)$ and $\langle C \rangle * B = \widehat{0}$. It suffices to show that $C \cdot S \equiv \mathsf{zeroPoly}$, meaning that for any variable assignment A, we should obtain $\mathsf{eval}(C \cdot S, A) = 0$. Let M' be the length m vector obtained by applying eval to the monomial sequence M with assignment A. Let E be the $r \times m$ matrix where entry (i, j) is $B(i, j) * (M')_j \bmod p$. Thus the sum of row i, $\mathsf{sum}(E_i) = \mathsf{eval}(S_i, A)$. Let M^* be the $m \times m$ diagonal matrix in which $M^*(i, i) = M'_i$, with zeroes for off-diagonal entries.

$$\begin{aligned}
\mathsf{eval}(C \cdot S, A) &= \mathsf{sum}(\langle C \rangle * E) \\
&= \mathsf{sum}(\langle C \rangle * (B * M^*)) \\
&= \mathsf{sum}((\langle C \rangle * B) * M^*) \\
&= \mathsf{sum}(\widehat{0} * M^*) \\
&= \mathsf{sum}(\widehat{0}) \\
&= 0
\end{aligned}$$

□

A typical application of this will be to model something by a canonical polynomial sequence of length r, with m monomials. With modDim, to show $r \leq m$, we just need to show the polynomial sequence is linearly independent. Thus we will want to obtain canonical polynomial sequences. For our application we in fact naturally begin with such a sequence, and then perform standard operations like $\mathsf{polyProd}$ which preserve canonicity.

The proof of RCW

The original proof of the RCW theorem due to [19] used higher incidence matrices. We formalize a simplified proof due to [3], based on polynomial vector spaces. We need to assume at least $\binom{n}{\|L\|}$ exists to state it, but we assume a little more, that $n^{\|L\|}$ exists (recall by lemma 2.4.4 this implies that $\binom{n}{\|L\|}$ exists).

Theorem 5.2.38 *(RCW theorem formalized)* $\mathrm{V}^0 + \mathsf{enum} + \mathsf{modDim} + \Delta_1^{1,\mathrm{b}} - \mathsf{CA}$ *proves that if p is a prime, and both $n^{\|L\|}$ and $p^{\|L\|}$ exist, then:*

$$\mathsf{isIncidenceType}(H, m, n, \overline{(L \bmod p)}, (L \bmod p)) \Rightarrow m \leq \binom{n}{\|L\|} + \binom{n}{\|L\| - 1} + \ldots + \binom{n}{0}.$$

Note that if L has standard size then we already know $n^{\|L\|}$ exists. The use of non-standard size will be used by the Frankl and Wilson bound in the next chapter. In the typical case $p \leq n$ so

existence of $p^{\|L\|}$ is guaranteed, but for some applications we don't want to worry about whether or not $p \leq n$, so we explicitly require the existence of $p^{\|L\|}$. By the sum $\binom{n}{\|L\|} + \binom{n}{\|L\|-1} + \ldots + \binom{n}{0}$ we mean to define a length $\|L\| + 1$ vector of numbers and then sum it (we can *not* do this with numbers coding sequences since knowing $n^{\|L\|}$ exists only allows sequences with elements whose size are about n).

First we sketch the informal proof. Suppose $\mathbf{H} = \{H_0, \ldots, H_{m-1}\}$ is our set system on ground set $[n]$. For $i \neq j < m$ $\|H_i \cap H_j\| \in L \pmod p$ and for any $i < m$ $\|H_i\| \notin L \pmod p$. For each $H_i \subseteq [n]$ we define a polynomial $\mathbf{P}_i$ on variables $\vec{x} = x_0, \ldots, x_{n-1}$. Let $\mathbf{P}_i$ be:

$$\prod_{a \in L} (\vec{x} \cdot H_i - a),$$

where in $\vec{x} \cdot H_i$ we view H_i as a length n binary string and take the dot product (example: $\langle x_3, x_2, x_1, x_0 \rangle \cdot \langle 0, 1, 1, 0 \rangle$ is $x_2 + x_1$). We now multiply out each $\mathbf{P}_i$ to get $\widehat{\mathbf{P}}_i$, a sum of monomial terms. We then make a genuine modification to the $\widehat{\mathbf{P}}_i$ by multi-linearizing them, arriving at $\widetilde{\mathbf{P}}_i$. While the $\mathbf{P}_i$ and $\widehat{\mathbf{P}}_i$ are equal for all the variable assignments from $\mathbb{F}_p$ to the variables $x_0, \ldots, x_{n-1}$, the $\widetilde{\mathbf{P}}_i$ are only guaranteed to be equal for $0, 1 \in \mathbb{F}_p$. However this is enough to obtain the linear independence of the $\widetilde{\mathbf{P}}_i$. Now since the degree of the $\widetilde{\mathbf{P}}_i$ is at most $\|L\|$, they are contained in a space of polynomials with basis of size $\binom{n}{\|L\|} + \binom{n}{\|L\|-1} + \ldots + \binom{n}{0}$, thus obtaining the bound.

Now we give the formal proofs. Let s abbreviate $\binom{n}{\|L\|} + \binom{n}{\|L\|-1} + \ldots + \binom{n}{0}$. Fix an appropriate set system H. For each set H_i, for $i < m$, we want a polynomial that looks like $\mathbf{P}_i$. We will define a simple tree P, so that P_i will be a polynomial sequence whose product will correspond to $\mathbf{P}_i$. Let

$$\mathsf{t} := \; m — \|L\| — 2 \begin{array}{l} \diagup \; n+1 — n — 1 \\ \diagdown \; n+1 — p \end{array}$$

Let F enumerate the 1's of $L \subseteq [n+1]$. P_i will correspond to $\prod_{a \in L}(\vec{x} \cdot H_i - a)$ and $(P_i)_r$ corresponds to $(\vec{x} \cdot H_i - F(r))$. All our polynomials will be based on the single length $n+1$ monomial sequence $\langle 1, x_{n-1}, \ldots, x_0 \rangle$. Thus polynomial $(P_i)_r$ consists of $n+1$ monomials where monomial $k < n$ is the single variable x_k and monomial n is just the constant 1. The monomial sequence is independent of which polynomial and polynomial sequence we are in. So $P_{\langle i,r,0 \rangle}$ refers to the length $n+1$ monomial sequence, where the k^{th} monomial $P_{\langle i,r,0,k \rangle}$ is of the form $x_{n-1}^{d_{n-1}} \ldots x_0^{d_0}$, where at most one of $d_{n-1}, \ldots, d_0$ may equal 1, the rest being zero, so we have the bound of 1 on the exponent of a variable. Using F as a parameter, we can give a $\Sigma_0^{1,b}$ simple tree description of P, as follows:

$$P_{\langle m,r,1,k \rangle} := \begin{cases} -F(r) & \text{if } k = n \\ 1 & \text{if } k < n \text{ and } H_{\langle m,e \rangle} = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$P_{\langle m,r,0,k,e \rangle} := \begin{cases} 1 & \text{if } k = e \\ 0 & \text{otherwise} \end{cases}$$

We now want to multiply out each of these m polynomial sequences P_i, for $i < m$, to obtain the polynomials $\widehat{P}_i$; thus from P, the sequence of polynomial sequences, we will define $\widehat{P}$, a polynomial sequence. We define structure tree $\mathsf{s} :=$

$$m\text{—}2\begin{cases}(n+1)^{\|L\|}\text{—}n\text{—}\|L\| \\ (n+1)^{\|L\|}\text{—}p\end{cases}$$

We define $\widehat{P}$ to be the $\Sigma_0^{1,b}$ simple tree described by

$$(\widehat{P})_i := \mathsf{polyProd}(P_i).$$

All our polynomials will eventually relate back to evaluating on P, so given a variable assignment A (that is $\mathsf{isVector}(A, n, p)$), by $E^{(A)}$ we mean the length m vector such that $(E^{(A)})_i := \mathsf{product}((P_i)^{(A)})$. Since products respect evaluation (lemma 5.2.30), $\mathsf{eval}(\widehat{P}_i, A) = (E^{(A)})_i$. This is allowed because $n^{\|L\|}$ and p^b exist, where b is the degree of $\widehat{P}_i$ (note that $b \leq \|L\|$ so the existence of p^b follows from the existence of $p^{\|L\|}$). Since for $j < m$, H_j is a 0/1 vector (so $\mathsf{isVector}(H_j, n, p)$ holds), we can evaluate the polynomials at assignment H_j. Consider $\mathsf{eval}(\widehat{P}_i, H_j)$. If $i \neq j$, then it will evaluate to 0; if $i = j$ it will evaluate to a non-zero number.

> If $i \neq j$ then $\|H_i \cap H_j\| \in L \ (\mathsf{mod}\ p)$; let $a \in L$ be a number such that $\|H_i \cap H_j\| = a \ (\mathsf{mod}\ p)$. Thus an element in the sequence $P_i^{(H_j)}$ corresponds to evaluating $(H_j \cdot H_i - a) = (\|H_j \cap H_i\| - a) = 0 \ (\mathsf{mod}\ p)$. A product with a zero entry is zero, so $\mathsf{eval}(\widehat{P}_i, H_j) = \mathsf{product}(P_i^{(H_j)}) = 0$.
>
> If $i = j$ then every element of $P_i^{(H_j)}$ corresponds to an expression $(H_i \cdot H_i - a)$, where $\|H_i\| = H_i \cdot H_i \neq a \ (\mathsf{mod}\ p)$, so we have a product of non-zero numbers, thus $\mathsf{eval}(\widehat{P}_i, H_j) = \mathsf{product}(P_i^{(H_j)}) \neq 0$.

We now multi-linearize, letting $(\widetilde{P})_i := \mathsf{MLize}(\widehat{P}_i)$. As it stands $\widetilde{P}$ has a monomial sequence of length $(n+1)^{\|L\|}$, the same as $\widehat{P}$. To get a bound improvement we will consolidate $\widetilde{P}$ relative to the monomial sequence M. By M we mean the sequence of $\binom{n}{\|L\|} + \ldots + \binom{n}{0}$ multilinear monomials with $\leq \|L\|$ variables (of power 1) per a monomial. A key point is that this monomial sequence M can be defined by use of the function $\mathsf{setNumber}$ which we showed we could work with in this context. We then consolidate $\widetilde{P}$ with respect to M, letting $D_i := \mathsf{consolidate}(\widetilde{P}_i, M)$. Using lemmas 5.2.34 and 5.2.32 we have that for any binary vector H_j,

$$\mathsf{eval}(D_i, H_j) = \mathsf{eval}(\widetilde{P}_i, H_j) = \mathsf{eval}(\widehat{P}_i, H_j) = E^{(H_j)}.$$

Now we want to apply modDim to D. D is a canonical polynomial sequence. Now we show $\mathsf{linIndepPoly}(D)$. Consider any C such that $\mathsf{isVector}(C, m, p)$ and suppose $C \cdot D \equiv \mathsf{zeroPoly}$. For any $i < m$,

$$0 = \mathsf{eval}(C \cdot D, H_i) = C \cdot E^{(H_i)} = aC_i,$$

where $a \neq 0$, thus C is the zero vector $\widehat{0}$. Since D is a canonical, linearly independent polynomial sequence, the $m \times s$ matrix $\mathsf{CMatrix}(D)$ is linearly independent so by modDim, $m \leq s = \binom{n}{\|L\|} + \binom{n}{\|L\|-1} + \ldots + \binom{n}{0}$), finishing the proof.

Then we get the corollary to the RCW theorem, also used in the Frankl and Wilson bound. A key assumption below is the existence of a prime larger than an arbitrary number. While provable in S_2, this question is open for $\mathrm{I}\Delta_0$ and so also for V^0. In our application later we will have such a prime.

Corollary 5.2.39 $\mathrm{V}^0 + \mathit{enum} + \mathit{modDim} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$ *proves that if* $n^{\|L\|}$ *exists and there exists a prime* p *larger than* $k \notin L$*, such that* $p^{\|L\|}$ *exists then:*

$$\mathit{isIncidenceType}(H, m, n, \{k\}, L) \Rightarrow m \leq \binom{n}{\|L\|} + \binom{n}{\|L\| - 1} + \ldots + \binom{n}{0}.$$

Proof

Since $p > k$, where k is the size of a set in the set system, L is the same as $L \pmod p$, so intersections are in $L \pmod p$. Since $k \notin L$, no set size is in $L \pmod p$. Now we apply the above theorem.

□

Chapter 6

Ramsey Theory in Bounded Arithmetic

We touched on Ramsey theory in chapter 3, formalizing the probabilistic method to show that $S^1_2 + \mathsf{rWPHP}(\Sigma^b_1)$ proves a formalization of $R_2(k) > 2^{k/2}$. While that proof was non-constructive, we will formalize some constructive Ramsey lower bounds in section 6.3. In section 6.4 we will translate the result from the first order context into the second order context so that they can all be compared. In section 6.1 we will discuss Ramsey upper bounds, improving the bounds given by Pudlák. In section 6.2 we consider some reversals, showing that some Ramsey upper bound statements imply the principles used to prove them.

6.1 Ramsey Upper Bounds

Pudlák [40] showed that bounded arithmetic can prove a formalization of the upper bound of $R_r(k) \leq r^{rk}$. First we improve this bound to $R_r(k) \leq (1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r} \approx r^{rk}\frac{(1+\epsilon)}{r^{r-1/2}(k-1)^{(r-1)/2}\sqrt{2\pi}^{r-1}}$. The last approximate equality uses Stirling's formula, a form of which can be proven in S^1_2 by the work of [29, appendix]. Note that for the special case of $r = 2$, we get $R_2(k) \leq (1+\epsilon)\binom{2k-2}{k-1}$, which is about the best known even for this special case. However, for the special case of $k = 3$, this bound yields only $R_r(3) \leq (1+\epsilon)\frac{(2r)!}{2^r}$. We provide a special argument catered to this case that improves this further to $R_r(3) \leq 3(1+\epsilon)(r!)$, which is about the best known.

Throughout this section we will formalize the statements of the form $\mathsf{Ramsey}(\mathtt{H}, n \to (k)_r)$, where $\mathtt{H}$ is a three place relation symbol (recall definition 3.1.1). Thus we will work in the theories relativized to $\mathtt{H}$.

6.1.1 General Improvement

Now we consider the general case, showing the following.

Theorem 6.1.1 *There is a $\Sigma^b_2(\mathtt{H})$ formula $\psi(\mathtt{H})$ such that $\mathrm{S}^2_2(\mathtt{H}) + \mathit{fWPHP}(\psi(\mathtt{H}))$ proves*

$$\mathit{Ramsey}(\mathtt{H}, (1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r} \to (k)_r).$$

First we note how this theorem leads to a corollary.

Corollary 6.1.2 *$\mathrm{T}^4_2(\mathtt{H})$ proves $\mathit{Ramsey}(\mathtt{H}, (1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r} \to (k)_r)$.*

Proof

By theorem 6.1.1, it suffices to show that $\mathrm{T}^4_2(\mathtt{H})$ proves $\mathsf{fWPHP}(\psi(\mathtt{H}))$ for ψ being $\Sigma^b_2(\mathtt{H})$. We know that $\mathrm{T}^2_2(\mathtt{R})$ proves $\mathsf{fWPHP}(\mathtt{R})$, so for $\psi(\mathtt{H})$ replacing $\mathtt{R}$ in the $\Sigma^b_2(\mathtt{R})$ induction axioms, we can push negations in and pull out quantifiers to see that it suffices to have $\Sigma^b_4(\mathtt{H})$ induction.

□

We now prove theorem 6.1.1, following Pudlák's argument (we also follow the presentation in Krajíček [33], theorem 12.1.3).

Assuming the Ramsey principle is false allows us to define an injection from the the set of vertices to a smaller set of sequences. Such an injection will violate the fWPHP, giving us our desired contradiction. Our set of vertices is $[(1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r}]$, which we will map to the set of sequences with elements from $[r]$ and no more than $k-2$ of any number (let's call such sequences "good"). To this point the argument will be the same as Pudlák's. We shall then diverge by mapping the good sequences to $\frac{(rk-r)!}{((k-1)!)^r}$, so the composition of these two maps is an injection from $(1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r}$ to $\frac{(rk-r)!}{((k-1)!)^r}$, violating $\mathsf{fPHP}^{(1+\epsilon)n}_n$. We formalize the notion of good sequences as follows.

Definition 6.1.3 *For small $a_0, \ldots, a_{r-1}$ and r, let $\mathit{Good}_{a_0,\ldots,a_{r-1}}(x)$ be a Δ^b_1 (in S^1_2) formula expressing that x is a sequence with elements from $[r]$ having at most a_i i's.*

Since the parameters are small, the counting can be carried out by asserting the existence of small sets, so Σ^b_1 suffices. Since we can also express the predicate by a provably (in S^1_2) equivalent

Π^b_1 formula, the predicate's complexity is Δ^b_1. For convenience we will use the informal notation $\mathsf{Good}_{a_0,\ldots,a_{r-1}}$ to mean the set of all strings x such that $\mathsf{Good}_{a_0,\ldots,a_{r-1}}(x)$. When we refer to functions as having the set $\mathsf{Good}_{a_0,\ldots,a_{r-1}}$ as its range, we really mean that any value y the function takes on makes $\mathsf{Good}_{a_0,\ldots,a_{r-1}}(y)$ hold. A similar point holds for the domain of a function.

First, working in $\mathrm{S}^2_2(\mathtt{H})$, we shall repeat Pudlák's argument, describing the first map, from the set of vertices, $(1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r}$, to the set of good sequences, $\mathsf{Good}_{k-2,\ldots,k-2}$, under the assumption that $\mathtt{H}$ is an r-coloring of the edges with no size k monochromatic set. To aid the process we define a $\Pi^b_1(\mathtt{H})$ relation E with 2 sequences as arguments. The first is a sequence of vertices; the second is a sequence of colors (i.e. numbers from $[r]$).

$E(\langle x_0,\ldots,x_h\rangle,\langle\delta_0,\ldots,\delta_{h-1}\rangle)$ holds if

$$\begin{array}{ll} x_0 = 0 & \wedge \\ x_0 < x_1 < \ldots < x_h & \wedge \\ \forall i < j \leq h\ \mathtt{H}(x_i, x_j, \delta_i) & \wedge \\ \forall i < h \forall y < x_{i+1} \wedge y > x_i\ \exists j \leq i\ \neg\mathtt{H}(x_j, y, \delta_j) & \end{array}$$

In words, E says that we start with $x_0 = 0$ and let x_1 be the smallest numbered vertex such that edge $\{x_0, x_1\}$ is colored δ_0; x_2 is the next smallest such that edge $\{x_1, x_2\}$ is colored δ_1 and edge $\{x_0, x_2\}$ is colored δ_0, and so on. Given $x < (1+\epsilon)\frac{(rk-r)!}{((k-1)!)^r}$ we define $F(x)$ to be the unique sequence $\overline{\delta}$ such that for some $\langle x_0,\ldots,x_h\rangle$ we have $E(\langle x_0,\ldots,x_h\rangle,\overline{\delta})$, where $x_h = x$. We can show, using $\Sigma^{\mathrm{b}}_2(\mathtt{H})$−$\mathsf{LIND}$, that F is a well-defined, injective function; we only need length induction because the pertinent parameter in the inductive proofs is the length of the sequences, and such lengths are small. To show its range is indeed $\mathsf{Good}_{k-2,\ldots,k-2}$ consider what would happen if $\overline{\delta}$ had any color repeated $k-1$ times, at say $\delta_{i_1} = \ldots = \delta_{i_{k-1}}$, for $i_1 < \ldots < i_{k-1} < h$. Then the edges of $\{x_{i_1},\ldots,x_{i_{k-1}},x_h\}$ would all be colored δ_{i_1}, thus yielding a size k monochromatic set, violating our assumption. For the formula $\psi(\mathtt{H}, x, \overline{\delta})$ called for in the theorem, we take the $\Sigma^b_2(\mathtt{H})$ definition of $F(x) = \overline{\delta}$, namely $\exists\langle x_0,\ldots,x_h\rangle\ x_h = x \wedge E(\langle x_0,\ldots,x_h\rangle,\overline{\delta})$.

Now we define the map f, from $\mathsf{Good}_{k-2,\ldots,k-2}$ to $\frac{(rk-r)!}{((k-1)!)^r}$. We will refer to general small parameters as this facilitates the inductive arguments, and then in the end we will substitute $k-2$ for these parameters. For small parameters, let $G(a_0,\ldots,a_{r-1}) = \frac{(a_0+\ldots+a_{r-1}+r)!}{(a_0+1)!\ldots(a_{r-1}+1)!} - 1$, an approximation of the size of $\mathsf{Good}_{a_0,\ldots,a_{r-1}}$; the particular expression allows for the calculations to work out easily. An inductive proof shows that $G(a_0,\ldots,a_{r-1})$ is an integer, which facilitates the argument since the fWPHP deals with functions on integers. If any $a_i = -1$ we define $G(a_0,\ldots,a_{r-1}) = 0$. We can see that for $a_i \geq 0$, G satisfies the following recursive bound.

$$\begin{aligned} G(a_0,\ldots,a_{r-1}) \;\geq\; & 1 + G(a_0-1,a_1,\ldots,a_{r-1}) \\ & +G(a_0,a_1-1,a_2,\ldots,a_{r-1}) \\ & +\ldots \\ & +G(a_0,\ldots,a_{r-2},a_{r-1}-1) \end{aligned}$$

To avoid confusion, note that for what follows, at most one parameter among any particular list $a_0,\ldots,a_{r-1}$ may have the number 1 subtracted from it. We define the function $f_{a_0,\ldots,a_{r-1}}$ (a function from $\mathsf{Good}_{a_0,\ldots,a_{r-1}}$ to $[G(a_0,\ldots,a_{r-1})]$) with the following recursive formulas (Notation: though in a first order context we reuse " $\frown$ " for concatenation and $\langle\rangle$ for the empty sequence).

Definition 6.1.4

- $f_{a_0,\ldots,a_{r-1}}(\langle\rangle) = \quad 0$
- $$\begin{aligned} f_{a_0,\ldots,a_{r-1}}(x \frown i) = \quad 1 \;&+\; f_{a_0,\ldots,a_i-1,\ldots,a_{r-1}}(x) \\ &+\; G(a_0-1,a_1,\ldots,a_{r-1}) \\ &+\; \ldots \\ &+\; G(a_0,\ldots,a_{i-1}-1,\ldots,a_{r-1}) \end{aligned}$$

Note that the definition of $f_{a_0,\ldots,a_{r-1}}$ is Σ_1^b since we only require a recursion with $a_0+\ldots+a_{r-1}$ steps (a small number), which can be carried out using a short sequence that can be coded by a number.

Claim 6.1.5 *For $a_0,\ldots,a_{r-1} \leq k-2$, and any sequence x such that $x \in \mathsf{Good}_{a_0,\ldots,a_{r-1}}$, we have $f_{a_0,\ldots,a_{r-1}}(x) < G(a_0,\ldots,a_{r-1})$.*

Proof

This proof is by induction on the length of the sequence x. To carry this out we use $\mathrm{sqbd}(s,l)$ to indicate a function whose value is larger than any number x, where x codes a sequence of length $\leq l$ whose elements are values $\leq s$ (developed in [9]). Let $\phi(d)$ be the following Π_1^b formula.

$$\forall x < \mathrm{sqbd}(r,(k-2)r)\forall a < \mathrm{sqbd}(k-2,r)\ \mathrm{len}(x) = d \wedge \mathsf{Good}_a(x) \wedge y = f_a(x) \Rightarrow y < G(a)$$

Working in S_2^1 we can use Π_1^b−LIND on ϕ. We indicate how the inductive step works. Consider some allowed x and a where $\text{len}(x) = d$ and the last element in the sequence x is i; so $x = x' \frown i$.

$$\begin{aligned} f_{a_0,\ldots,a_{r-1}}(x) &= 1 + f_{a_0,\ldots,a_i-1,\ldots,a_{r-1}}(x') + G(a_0 - 1, a_1, \ldots, a_{r-1}) + \\ &\quad \ldots + G(a_0, \ldots, a_{i-1} - 1, \ldots, a_{r-1}) \\ &< 1 + G(a_0 - 1, a_1, \ldots, a_{r-1}) + \\ &\quad \ldots + G(a_0, \ldots, a_i - 1, \ldots, a_{r-1}) \\ &\leq G(a_0, \ldots, a_{r-1}) \end{aligned}$$

The inductive hypothesis justifies the first inequality since $\text{len}(x') = d - 1$. The second inequality follows from the recurrence on G. Induction yields $\forall d\ \phi(|d|)$. We can find d such that $|d| = a_0 + \ldots + a_{r-1}$, since all the parameters are small, so we are done.

□

Claim 6.1.6 *For any $x, y \in \mathsf{Good}_{a_0,\ldots,a_{r-1}}$, such that $x \neq y$, $f_{a_0,\ldots,a_{r-1}}(x) \neq f_{a_0,\ldots,a_{r-1}}(y)$.*

Proof

We proceed by induction on the sequence lengths of x or y, using a Π_1^b formula similar to the one in the last proof. We point out how the inductive step works. Suppose $x = x' \frown i$ and $y = y' \frown j$. If $i = j$ then once we use the definition of f, we can apply the inductive hypothesis. If $i \neq j$, assume $i < j$, and then we can calculate.

$$\begin{aligned} f_{a_0,\ldots,a_{r-1}}(x' \frown i) &= 1 + f_{a_0,\ldots,a_i-1,\ldots,a_{r-1}}(x') + G(a_0 - 1, a_1, \ldots, a_{r-1}) + \\ &\quad \ldots + G(a_0, \ldots, a_{i-1} - 1, \ldots, a_{r-1}) \\ &< 1 + G(a_0 - 1, a_1, \ldots, a_{r-1}) + \\ &\quad \ldots + G(a_0, \ldots, a_i - 1, \ldots, a_{r-1}) \\ &\leq f_{a_0,\ldots,a_{r-1}}(y' \frown j) \end{aligned}$$

The first inequality follows from claim 6.1.5 and last from the definition of f.

□

Now we turn back to the particular case of $a_0 = \ldots = a_{r-1} = k - 2$. The function $f_{k-2,\ldots,k-2}$ is injective by claim 6.1.6. From claim 6.1.5 we see that it has the desired the range of $\frac{(rk-r)!}{((k-1)!)^r}$. And so we have finished approximately counting the set of good sequences.

6.1.2 Special Case: $k = 3$

We now consider the special case of $k = 3$. It is already covered in the above case by counting the set $\mathsf{Good}_{1,\ldots,1}$, sequences with elements from $[r]$, having no repeated elements. We will now count this set more efficiently, bounding its size by $3(r!)$, which yields the following theorem.

Theorem 6.1.7 *There is a $\Sigma^b_2(\mathtt{H})$ formula $\psi(\mathtt{H})$ such that $\mathrm{S}^2_2(\mathtt{H}) + \mathit{fWPHP}(\psi(\mathtt{H}))$ proves*

$$\mathit{Ramsey}(\mathtt{H}, 3(1+\epsilon)(r!) \to (3)_r).$$

The following corollary is proved in a similar way to corollary 6.1.2.

Corollary 6.1.8 *$\mathrm{T}^4_2(\mathtt{H})$ proves $\mathit{Ramsey}(\mathtt{H}, 3(1+\epsilon)(r!) \to (3)_r)$.*

To prove theorem 6.1.7, we proceed as in the proof of theorem 6.1.1, working in $\mathrm{S}^2_2(\mathtt{H})$ to map $3(1+\epsilon)(r!)$ to $\mathsf{Good}_{1,\ldots,1}$. Now we carry out the counting, describing a mapping ρ from $\mathsf{Good}_{1,\ldots,1}$ to $3(r!)$. The rest of the argument can be carried out in $\mathrm{S}^1_2(\mathtt{H})$. We will use $(r)_m$ to denote the product $r(r-1)\ldots(r-m+1)$.

Let $\rho(\langle a_1, \ldots, a_k\rangle) = (r)_0 + (r)_1 + \ldots + (r)_{k-1} + g_r(\langle a_1, \ldots, a_k\rangle)$; we will define g_s momentarily. We assume that $a_1 > a_2 > \ldots > a_k$ since if not we can define a map in S^1_2 that rearranges it as such. Now we define g_s recursively as follows.

- $g_s(\langle\rangle) = 0$
- $g_s(\langle a_1, \ldots, a_k\rangle) = (a_1)(s-1)_{(k-1)} + g_{s-1}(\langle a_2, \ldots, a_k\rangle)$

The definition of ρ is Σ^b_1 since the recursion is defined on short sequences. To show ρ is injective with proper range it suffices to prove the following two claims about g. Both proofs use induction on the sequence length, similar to the last subsection, but in this case $\Pi^b_1\text{-}\mathsf{LIND}$ suffices since we need not refer to the Σ^b_1 formula Good (as was done for the formula $\phi(d)$ in claim 6.1.5).

Claim 6.1.9 *$g_s(\langle a_1, \ldots, a_k\rangle) < (s)_k$, for $a_i \leq s - i$.*

Proof

Proceed by induction on k, the length of the sequence.

$$\begin{aligned} g_s(\langle a_1, \ldots, a_k\rangle) &= (a_1)(s-1)_{k-1} + g_{s-1}(\langle a_2, \ldots, a_k\rangle) \\ &< (a_1)(s-1)_{k-1} + (s-1)_{k-1} \\ &= (a_1+1)(s-1)_{k-1} \\ &\leq s(s-1)_{k-1} \\ &= (s)_k \end{aligned}$$

□

Claim 6.1.10 *For any small k and s, g_s is injective on length k sequences $\langle a_1, \ldots, a_k\rangle$ such that $a_i \leq s - i$.*

Proof

We use induction on k.

For $\langle a_1, \ldots, a_k\rangle \neq \langle b_1, \ldots, b_k\rangle$, we will show that $g_s(\langle a_1, \ldots, a_k\rangle) \neq g_s(\langle b_1, \ldots, b_k\rangle)$. If $a_1 = b_1$, then we have:

$g_s(\langle a_1, \ldots, a_k\rangle) = (a_1)(s-1)_{k-1} + g_{s-1}(\langle a_2, \ldots, a_k\rangle)$ and
$g_s(\langle b_1, \ldots, b_k\rangle) = (b_1)(s-1)_{k-1} + g_{s-1}(\langle b_2, \ldots, b_k\rangle)$.

By the inductive hypothesis $g_{s-1}(\langle a_2, \ldots, a_k\rangle) \neq g_{s-1}(\langle b_2, \ldots, b_k\rangle)$, finishing this case.

Now consider the case of $a_1 \neq b_1$; assume $a_1 < b_1$ and we show $g_s(\langle a_1, \ldots, a_k\rangle) < g_s(\langle b_1, \ldots, b_k\rangle)$.

$$\begin{aligned} g_s(\langle a_1, \ldots, a_k\rangle) &= (a_1)(s-1)_{k-1} + g_{s-1}(\langle a_2, \ldots, a_k\rangle) \\ &< (a_1)(s-1)_{k-1} + (s-1)_{k-1} \\ &= (a_1+1)(s-1)_{k-1} \\ &\leq (b_1)(s-1)_{k-1} \\ &\leq (b_1)(s-1)_{k-1} + g_{s-1}(\langle b_2, \ldots, b_k\rangle) \\ &= g_s(\langle b_1, \ldots, b_k\rangle) \end{aligned}$$

□

Now to check that the range is correct we use the following bound (note that we use 3, rather than e since it allows for a simple inductive proof in bounded arithmetic).

Lemma 6.1.11 S^1_2 *proves that for small r,* $(1/0! + 1/1! + \ldots + 1/r!) < 3$.

Proof

We show by induction on r show that $(1/0! + 1/1! + \ldots + 1/r!) < 3 - 2/(r+1)!$ For the inductive step, assuming the above, consider $(1/0! + 1/1! + \ldots + 1/r! + 1/(r+1)!)$. Using the inductive hypothesis, we bound the sum by $3 - 2/(r+1)! + 1/(r+1)! = 3 - 1/(r+1)! \leq 3 - 2/(r+2)!$

This argument goes through in S^1_2 since the induction is on a Σ^b_1 formula expressing the existence of a sum along with a bound.

□

So the range is $[3(r!)]$ since

$$\begin{aligned} \rho(\langle a_1, \ldots, a_r \rangle) &< (r)_0 + \ldots + (r)_r \\ &= r!(1/0! + 1/1! + \ldots 1/r!) \\ &< 3(r!). \end{aligned}$$

The last inequality follows from lemma 6.1.11 finishing the proof.

In all the cases, the Ramsey principles can be proved in $\mathrm{T}^4_2(\mathtt{H})$. Chiari and Krajíček [10] show that a Ramsey principle cannot be proved in $\mathrm{S}^2_2(\mathtt{H})$; the same kind of argument can be applied to all our Ramsey principles. Thus it is an open question as to where exactly the Ramsey principles fit in the hierarchy $\mathrm{S}^1_2(\mathtt{H}) \subseteq T^1_2(\mathtt{H}) \subseteq S^2_2(\mathtt{H}) \subseteq T^2_2(\mathtt{H}) \subseteq \ldots \subseteq S_2(\mathtt{H})$. This will be further discussed at the end of the next section.

6.2 The Ramsey Reversals

We will consider two reversals spending the bulk of the section on the first one. In the first reversal we show how the Ramsey principle (for $k = 3$) implies the fWPHP. We sketch the idea behind the proof now. Suppose for contradiction that fPHP^{2n}_n does not hold; in fact we can replace $2n$ by a larger quantity, $q(n)$ (to be defined). So we have an injective function from $q(n)$ to n. Assume that $n = 2^r$ for some r. We constructively exhibit an r-coloring of the graph on n vertices with no size 3 monochromatic set. Using our injective function we pull this coloring back to an isomorphic r-coloring for the graph on $q(n)$ vertices. Since the function is injective, this new coloring also has no size 3 monochromatic set. We in fact have that $q(n) > r^{3r}$, so since $(r^{3r} \to (3)_r)$ holds, we know there is a size 3 monochromatic set. We have arrived at a desired contradiction.

To prove our reversal we will formalize the constructive lower bound of $R_r(3) > 2^r$. The proof, as pointed out in [26, p.145] goes through easily by induction on r. For the inductive step we start with two $(r-1)$ colored graphs, each with 2^{r-1} vertices and no monochromatic sets of size 3. They are joined by edges of a new color, giving us the appropriate graph on 2^r vertices. The construction we give, based on this argument, essentially takes as its vertices the binary strings of length r, and

colors edges according to the first bit (from the right) at which two strings differ (recall that u_i refers to the i^{th} bit in the binary representation of u).

Definition 6.2.1 *Let* $\mathsf{Low}(x, y, k)$ *be the* Σ_0^b *formula:*

$$(\forall i < \mathsf{max}(|x|, |y|)\ (i < k \Rightarrow x_i = y_i)) \wedge x_k \neq y_k.$$

Now we can prove that Low really is a lower bound coloring.

Lemma 6.2.2 S_2^1 *proves* $\neg\mathsf{Ramsey}(\mathsf{Low}, 2^r \rightarrow (3)_r)$.

Proof

Fix some small r. S_2^1 proves for all $X = \{u, v, w\} \subseteq [2^r]$, $\neg\mathsf{monochromatic}(\mathsf{Low}, X, 2^r, r)$. Consider any color $d < r$, and we show that X is not colored just by d. If $u_d = v_d$ then edge $\{u, v\}$ is not colored by d, so assume $u_d \neq v_d$. Then w_d has to equal one of u_d or v_d, so not all the edges can be colored d.

We prove $\mathsf{coloring}(\mathsf{Low}, 2^r, r)$ by length induction up to r, noting that for $x < y < 2^r$, the first bit where they differ will be at a unique position $d < r$.

□

Definition 6.2.3 *For notational convenience let* $q(x) := r^{3 \log 2x}$.

Theorem 6.2.4 *Let* $i \geq 1$. *For any formula* $\phi(\mathtt{R})$ *of complexity* $\Sigma_i^b(\mathtt{R})$, *there is a formula* $\psi(\mathtt{R})$ *of complexity* $\Sigma_i^b(\mathtt{R})$ *such that*

$\mathrm{S}_2^1(\mathtt{R}) + \forall r\ \mathsf{Ramsey}(\psi(\mathtt{R}), r^{3r} \rightarrow (3)_r)$ *proves* $\forall n\ \mathsf{fPHP}_n^{q(n)}(\phi(\mathtt{R}))$.

Proof

Let $\psi(x_1, x_2, k)$ be the $\Sigma_i^b(\mathtt{R})$ formula:

$$\exists y_1, y_2 < 2^r\ (\phi(\mathtt{R}, x_1, y_1)\ \wedge\ \phi(\mathtt{R}, x_2, y_2)\ \wedge\ \mathsf{Low}(y_1, y_2, k)).$$

To show $\mathsf{fPHP}_n^{q(n)}(\phi(\mathtt{R}))$ fix some n and find the r such that $2^{r-1} \leq n < 2^r$; note that r is small.

Assume $\neg\mathsf{fPHP}_n^{q(n)}(\phi(\mathtt{R}))$, so $\phi(\mathtt{R})$ is a total injective function from $q(n)$ to n. Showing $\neg\mathsf{Ramsey}(\psi, r^{3r} \rightarrow (3)_r)$ finishes the proof, so it suffices to show the following 2 claims.

Claim 6.2.5 $\mathrm{S}^1_2(\mathtt{R})$ *proves* $\mathsf{coloring}(\psi, r^{3r}, r)$.

Claim 6.2.6 $\mathrm{S}^1_2(\mathtt{R})$ *proves* $\forall x_1 < x_2 < x_3 < r^{3r}\ \neg\mathsf{monochromatic}(\psi, \{x_1, x_2, x_3\}, r^{3r}, r)$.

To prove the first claim, consider any edge given by $x_1, x_2 < r^{3r}$. Note that $x_1 < r^{3r} = q(2^{r-1}) \leq q(n)$, therefore, x_1 is mapped by $\phi(\mathtt{R})$ to a number $y_1 < n < 2^r$. Similarly, x_2 is mapped to some $y_2 < 2^r$. By lemma 6.2.2, Low assigns a color to the edge (y_1, y_2), so (x_1, x_2) is also assigned a color.

To prove the second claim, for $j = 1, 2, 3$, let $y_j < 2^r$ be such that $\phi(\mathtt{R}, x_j, y_j)$. The set $\{y_1, y_2, y_3\}$ is not monochromatic by lemma 6.2.2, therefore neither is $\{x_1, x_2, x_3\}$, since the latter set is colored in the same manner as the former.

□

Now, theorem 6.2.4 together with theorem 2.3.4 immediately yields the following theorem, the reversal.

Theorem 6.2.7 (Reversal to weak pigeonhole principle) *Let $i \geq 1$, and $\phi(\mathtt{R})$ be a formula of complexity $\Sigma^b_i(\mathtt{R})$. Then there is a formula $\psi(\mathtt{R})$ of complexity $\Sigma^b_i(\mathtt{R})$ such that*

$$\mathrm{S}^1_2(\mathtt{R}) + \mathsf{Ramsey}(\psi(\mathtt{R}), r^{3r} \to (3)_r) \text{ proves } \mathsf{fWPHP}(\phi(\mathtt{R})).$$

Note a special case of particular interest if we take $\phi(\mathtt{R})$ to be simply $\mathtt{R}$; then the above theorem tells us that there is a $\Sigma^b_1(\mathtt{R})$ formula with the appropriate proof going through in $\mathrm{S}^1_2(\mathtt{R})$. From this fact we can obtain the following independence result.

Corollary 6.2.8 *There is a $\Sigma^b_1(\mathtt{R})$ formula $\psi(\mathtt{R})$ such that $S^2_2(\mathtt{R})$ does not prove*

$$\mathsf{Ramsey}(\psi(\mathtt{R}), r^{3r} \to (3)_r).$$

Proof

We take the ψ from theorem 6.2.7, for $i = 1$, so that if $\mathrm{S}^2_2(\mathtt{R})$ did prove the Ramsey statement, it would also prove $\mathsf{fWPHP}(\mathtt{R})$ (in fact $\mathtt{R}$ can be replaced by $\Sigma^b_1(\mathtt{R})$). However as mentioned earlier (see remarks following theorem 2.3.3), this is known to be unprovable in $\mathrm{S}^2_2(\mathtt{R})$.

□

However, the argument of Chiari and Krajíček [10] can be applied to give a proof of the following stronger result.

Theorem 6.2.9 *(essentially [10])* $S_2^2(\mathtt{H})$ *does not prove* $\mathsf{Ramsey}(\mathtt{H}, r^{3r} \to (3)_r)$.

The reversal of theorem 6.2.7 can be mildly improved upon, using basically the same argument and the same lower bound. As long as k (the size of the monochromatic set we are looking for) is a constant larger than 2 (it was 3 above), or even uniform in the statement, but polynomially bounded in r, the lower bound used for $k = 3$ is good enough to get the reversal. However other reversals of this kind seem difficult to obtain. The difficulty is apparent by considering two key ingredients for obtaining a reversal in this manner: 1) The lower bound is constructive and 2) the upper bound is not too much larger than the lower bound (for example, in the above argument we had $2^r < R_r(3) \leq r^{3r}$, and the bounds were related by the term $q(x)$ in the sense that $r^{3r} \leq q(2^r)$). If we consider the typical case of $R_2(k)$, we have an upper bound of 2^{2k}, but the best known constructive lower bound is about $R_2(k) \geq e^{(\log^2 k)/(\log\log k)}$ (see [25]). The bounds are too far apart to obtain a reversal by these methods.

We now consider a different kind of reversal (suggested by Stephen Simpson), obtained from a different Ramsey principle. Up to this point we have considered Ramsey principles in which the upper bound is explicitly given. Now we consider the case in which the appropriate number is only asserted to exist, namely $\forall r \exists n \mathsf{Ramsey}(\mathtt{H}, n \to (3)_r)$. This can easily be proven with the axiom EXP. The reversal is something like a converse, where the Ramsey principle is replaced by a schema (we put Σ_0^b in place of the relation symbol $\mathtt{H}$ to indicate that the principle holds for all Σ_0^b formulae).

Theorem 6.2.10 (Reversal to EXP) $S_2^1 + \forall r \exists n$ $\mathsf{Ramsey}(\Sigma_0^b, n \to (3)_r)$ *proves* EXP.

Proof

We actually only use the Ramsey principle for the formula Low (a Σ_0^b formula). Given r there is an n such that $\mathsf{Ramsey}(\mathsf{Low}, n \to (3)_r)$. To show EXP it suffices to show $r < |n|$. It is a general fact that for $b \geq a$, $\mathsf{Ramsey}(\mathtt{H}, n \to (k)_b)$ implies $\mathsf{Ramsey}(\mathtt{H}, n \to (k)_a)$. So if $r \geq |n|$, we would arrive at $\mathsf{Ramsey}(\mathsf{Low}, n \to (3)_{|n|})$, but we in fact have that $\neg\mathsf{Ramsey}(\mathsf{Low}, n \to (3)_{|n|})$ (basically by the argument of lemma 6.2.2). Thus $r < |n|$ as desired.

□

Though this result is not unexpected, note the significance of the constructive lower bound. It would seem difficult to prove related results with different Ramsey principles in which the constructive lower bounds are not good enough.

6.3 Constructive Lower Bounds

We now discuss constructive Ramsey lower bounds. We will begin by discussing our notation for Ramsey theory in the second order context. Then we will formalize increasingly better bounds. We will consider the well-known constructive Ramsey lower bound due to Frankl and Wilson [25] which uses linear algebra and can be formalized using the dimension principle. However first we will consider some weaker bounds which can be formalized without the dimension principle.

To formalize this we essentially use the adjacency matrix representation of a graph. Given an edge coloring (with r colors) of a complete graph with vertex set $[n]$, we can represent this as an $n \times n$ matrix (rows and columns are labeled by $[n]$ in increasing order) in which entry (i, j) is the color (i.e. a number from $[r]$) of the edge connecting vertices i and j; we ignore what happens on the diagonal entries (i, i). Notice that the matrix is symmetric. Now we can give the definition of the Ramsey arrow notation for second order logic, putting a "2" in the various notation to indicate second order logic (since we use the same names as used in the first order context).

Definition 6.3.1

- *Let* $\mathit{coloring2}(G, n, r)$ *be:*
  $\mathit{isMatrix}(G, n \times n, r) \ \wedge \ \forall i < j < n \ (G_i)_j = (G_j)_i$.
- *Let* $\mathit{monochromatic2}(H, X, n, r)$ *be:* $\exists d < r \ \forall u \neq v \in X \ ((G_u)_v) = d$.
- *Let* $\mathit{Ramsey2}(G, n \to (k)_r)$ *be*
  $\mathit{coloring2}(G, n, r) \Rightarrow \exists X \subseteq [n] \ (\mathit{size}(X) = k \wedge \mathit{monochromatic2}(G, X, n, r))$.

We consider the simple constructive lower bound of $R_2(k) > (k-1)^2$. To prove it we describe a 2-coloring, G, on $(k-1)^2$ vertices with no size k monochromatic set. Group the vertices into $(k-1)$ groups, each of size $(k-1)$. Make the colors of edges within a group red, and those between groups blue. A red monochromatic set must be contained in a single group, so has size $\leq (k-1)$. A blue monochromatic set can have at most one vertex per a group, so has size $\leq (k-1)$. We can formalize this as follows.

Proposition 6.3.2 $\mathrm{V}^0 + \mathit{PHP}$ *proves* $\exists G \ \neg \mathit{Ramsey2}(G, (k-1)^2 \to (k)_2)$.

Proof

We can give a $\Sigma_0^{1,b}$ description of matrix G according to the above description as follows. We can take the vertices $u < (k-1)^2$ to be pairs $\langle a, b \rangle$ with $a, b < k-1$, so group i $(i < k-1)$ is the set of pairs with a first entry i; we define G as described above so that

it is a coloring. Now suppose for contradiction that we have $X \subseteq [n]$, and F such that $\mathsf{CF}(F, k \to X)$. If X is monochromatic red (i.e. contains pairs all with the same first element), then we can inject X into the values of its second entry, giving us an injection $k \hookrightarrow (k-1)$, contradicting PHP. If X were monochromatic blue, every pair would have a different first element, so we could again inject $k \hookrightarrow (k-1)$.

□

We will now beat this bound, working in the same theory, though the proof is more complicated. Nagy [37] discovered a better constructive Ramsey lower bound, showing that $R_2(k) > \binom{k-1}{3}$. To prove this we describe a graph G with $\binom{k-1}{3}$ vertices, each vertex being a size 3 subset of $[k-1]$. For 2 vertices $u, v \subseteq [k-1]$, if $\|u \cap v\| = 1$ then color the edge between them red; otherwise the intersection is 0 or 2, and we color the edge blue. A monochromatic red set corresponds to a type $(\{3\}, \{1\}, k-1)$ set system and so has at most $k-1$ sets by the Fisher Inequality (in fact we can apply theorem 5.1.5 to this case). A monochromatic blue set corresponds to a type $(\{3\}, \{0,2\}, k-1)$ set system, so by the Oddtown theorem it has at most $k-1$ sets (in fact we can apply the special case of theorem 5.1.2). Thus no size k set is monochromatic. We can easily define this graph in V^0 and then the two results we cited use $\mathrm{V}^0 + \mathsf{PHP}$, thus we have the following theorem.

Theorem 6.3.3 $\mathrm{V}^0 + \mathit{PHP}$ *proves* $\exists G\ \neg \mathit{Ramsey2}(G, \binom{k-1}{3} \to (k)_2)$.

Recall that the only known proofs of the Fisher Inequality and the Oddtown theorem use linear algebra. The alternative proofs (i.e. avoiding linear algebra) we obtained for special cases is what allows us to formalize the above theorem using only $\mathrm{V}^0 + \mathsf{PHP}$.

Now we move on to Frankl and Wilson's constructive lower bound of $R_2(k) \geq e^{((1-\epsilon)\ln^2 k)/(4 \ln \ln k)}$; we will modify the bound, using "2" instead of "e" and $\log_2$ instead of ln (i.e. $\log_e$). The main ideas are contained in the coming lemma 6.3.6, from which theorem 6.3.7 will follow by a calculation. A key step in this calculation is Bertrand's postulate, which says that for any natural number n, there is a prime number p such that $n < p \leq 2p$. Paris, Wilkie, and Woods [39] showed that for some standard c, S_2 can prove that there is a prime between n and n^c, thus proving the infinitude of the primes. In $\mathrm{I}\Delta_0$ the infinitude of the primes is an open question. However D'Aquino showed the following special case of Bertrand's postulate, which suffices for our purposes.

Theorem 6.3.4 *(Bertrand's Postulate [14])* $\mathrm{I}\Delta_0$ *proves that if* 2^x *exists then*

$$(\exists p\ x < p \leq 2x) \wedge \mathit{prime}(p).$$

The following lemma gives a bound we will use.

Lemma 6.3.5 $\mathrm{V}^0 + \mathit{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$ *proves that if* $\binom{n}{k}$ *exists and* $k \leq (n/2)$*, then*

$$\binom{n}{k} + \binom{n}{k-1} + \ldots + \binom{n}{0} \leq 2\binom{n}{k}.$$

The lemma can be proved by showing the stronger claim of

$$\binom{n}{k} + \binom{n}{k-1} + \ldots + \binom{n}{0} \leq (1 + \frac{k}{n-2k+1})\binom{n}{k},$$

using induction on k.

Lemma 6.3.6 *We work in the theory* $\mathrm{V}^0 + \mathit{enum} + \mathit{modDim} + \Delta_1^{1,\mathrm{b}}\text{-}\mathit{CA}$*. Let* p *be a prime, and suppose* $\binom{p^3}{p^2-1}$ *exists. Then we can prove that:*

$$\exists G\ \neg \mathit{Ramsey}(G, \binom{p^3}{p^2-1} \to (2\binom{p^3}{p-1})_2)$$

First we discuss the informal proof. We construct the graph G with $\binom{p^3}{p^2-1}$ vertices, by letting its vertex set be the collection of size p^2-1 size subsets of $[p^3]$. For two distinct vertices $u, v \subseteq [p^3]$, we color the edge between them red if $\|u \cap v\| \neq p-1 \pmod p$ (i.e. it is equal $(\mathsf{mod}\ p)$ to one of $0, 1, \ldots, p-2$). We color the edge blue otherwise (that is, in the case that $\|u \cap v\| = p-1 \pmod p$).

Suppose X is a set of vertices of G which is monochromatic red. Then X is in fact a type $(\overline{L \pmod p}, L \pmod p, p^3)$ set system, where $L := \{0, 1, \ldots, p-2\}$ (notice that the sets are all of size $p^2 - 1 = p - 1 \pmod p$, so the sizes of the sets really are in $\overline{L \pmod p}$). Thus, by the RCW theorem,

$$\begin{aligned} \|X\| &\leq \binom{p^3}{p-1} + \binom{p^3}{p-2} + \ldots + \binom{p^3}{0} \\ &\leq 2\binom{p^3}{p-1}. \end{aligned}$$

The last inequality follows from lemma 6.3.5. Suppose X is monochromatic blue. Then X is a type $(\{p^2-1\}, J, p^3)$ set system, where $J := \{p-1, 2p-1, \ldots, (p-1)p-1\}$. By corollary 5.2.39 and lemma 6.3.5, $\|X\| \leq 2\binom{p^3}{p-1}$.

Now we give the formal proof. We have already set things up so that it will follow the informal proof closely, though we point out some issues particular to working in this weak theory.

Proof

Let p be some prime for which $\binom{p^3}{p^2-1}$ exists. We let the number of vertices $n := \binom{p^3}{p^2-1}$. We can think of a vertex $x < n$ as a subset of $[p^3]$ of size (p^2-1) via the inverse of the $\mathsf{setNumber}$ function (recall lemma 5.1.8), which we call $\mathsf{setNumber}^{-1}$. Thus we can $\Delta_1^{1,b}$ define the $n \times n$ 2-coloring G according to the above informal description. Now we prove that $\neg\mathsf{Ramsey}(G, \binom{p^3}{p^2-1} \to (2\binom{p^3}{p-1})_2)$.

Let $X \subseteq [n]$, such that X is monochromatic in G and let its size be m. We can view each element of X as a subset of $[p^3]$ of size p^2-1, thus from it we can define a corresponding H such that $\mathsf{isIncidenceType}(H, m, p^3, \{p^2-1\}, L)$, where L will depend on whether or not X is monochromatic red or blue. Note that by saying that H corresponds to X, we mean that $\mathsf{setNumber}^{-1}(X_i) = H_i$.

Now we break the argument into cases on whether the color of X is red or blue, applying a different linear algebra bound in each case. For both cases the formalized version needs $n^{\|L\|} = (p^3)^{p-1}$ to exist. It does because of lemma 2.4.3 which yields:

$$\binom{p^3}{p^2-1} \geq \left(\frac{p^3-p^2}{p^2-1}\right)^{p^2-1} \geq (p-1)^{p^2-1} \geq (p^3)^{p-1},$$

for sufficiently large p.

For the red case, we apply the formalized RCW theorem as in the above proof to obtain the bound (we of course also have the other size condition that p^{p-1} exists). For the blue case, we use the corollary to the RCW theorem following the informal proof. Note that for the formalized corollary we need to know that there exists a prime q larger than p^2-1. Since 2^{p^2} exists, by Bertrand's Postulate we can find such a prime $q \leq 2p^2$. Since $(2p^2)^{p-1}$ exists, so does $q^{\|L\|} \leq (2p^2)^{p-1}$, so we can apply the corollary with the prime q.

□

Now we prove the Frankl and Wilson theorem, which is essentially a corollary of lemma 6.3.6 and theorem 6.3.4. Notice that here we will actually need the full power of Bertrand's Postulate. The last proof used it, but in fact it would have been enough if there were a standard c such that for any number n there were a prime p such that $n < p \leq n^c$; in the following proof, just $c = 2$ would put us in a bind.

Theorem 6.3.7 *(Formalized Frankl and Wilson)* $\mathrm{V}^0 + \mathsf{enum} + \mathsf{modDim} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ *proves that if* $n := 2^{\left(\frac{\log^2 k}{2^8 \log\log k}\right)}$ *exists then* $\exists G\ \neg\mathsf{Ramsey}(G, n \to (k)_2)$.

Proof

Let p be the largest prime such that $2\binom{p^3}{p-1} < k$. By Bertrand's Postulate (theorem 6.3.4) there is a prime q such that $p < q \leq 2p$, thus $k < 2\binom{(2p)^3}{2p-1}$. Now we want to show that $\binom{p^3}{p^2-1}$ exists; it suffices by lemma 2.4.4 to show that p^{3p^2} exists. For a number of our calculations, the inequality will be true for sufficiently large p (that is, for $p > c$ for some standard c); the smaller cases can be checked by brute force, so the theorem will be true for all k. By a calculation

$$k > 2\binom{p^3}{p-1} \geq \left(\frac{p^3-(p-1)}{p-1}\right)^{p-1} \geq (p^2-1)^{p-1}.$$

Thus $\log k > (p-1)\log(p^2-1)$ and $\log\log k > \log(p-1)$. By a calculation,

$$k < 2\binom{(2p)^3}{2p-1} \leq ((2p)^3)^{2p} = 64^p p^{6p}.$$

Thus $\log k < 7p\log p$ and $\log\log k < 2\log p$.

$$\begin{aligned} 2^{\left(\frac{\log^2 k}{2^8 \log\log k}\right)} &> 2^{\left(\frac{(p-1)^2\log^2(p^2-1)}{2^8(2\log p)}\right)} \\ &> 2^{(1/2^9)(p-1)^2\log(p^2-1)} \\ &\geq p^{(1/2^9)(p-1)^2} \end{aligned}$$

The last number is big enough to obtain a number larger than p^{3p^2}, by a few arithmetic operations. Thus we have achieved our goal of showing that $\binom{p^3}{p^2-1}$ exists and can apply lemma 6.3.6 to obtain

$$\exists G\ \neg\mathsf{Ramsey2}(G, \binom{p^3}{p^2-1} \rightarrow (2\binom{p^3}{p-1})_2).$$

The coloring G has no size $2\binom{p^3}{p-1}$ monochromatic set and so no size k monochromatic set. To finish the proof we just need to show that we have enough vertices, that is

$$\binom{p^3}{p^2-1} \geq 2^{\left(\frac{\log^2 k}{2^8\log\log k}\right)},$$

or equivalently $\log\binom{p^3}{p^2-1} \geq (\log^2 k)/(2^8\log\log k)$. Our bounds yield the following:

$$\frac{\log k}{8\log\log k} \leq \frac{7p\log p}{8\log(p-1)} \leq p,$$

which we apply in the following calculation:

$$\begin{aligned}
\log \binom{p^3}{p^2-1} &\geq \log \left(\frac{p^3-p^2+1}{p^2-1} \right)^{p^2-1} \\
&\geq \log(p-1)^{p^2-1} \\
&= (p^2-1)\log(p-1) \\
&\geq (p^2 \log p)/2 \\
&\geq \left((1/2) \left(\frac{\log k}{8 \log\log k} \right)^2 \log \left(\frac{\log k}{8 \log\log k} \right) \right) \\
&\geq \left(\frac{\log^2 k}{2^7 (\log\log k)^2} \right) (1/2) \log\log k \\
&\geq \frac{\log^2 k}{2^8 \log\log k}
\end{aligned}$$

□

6.4 Comparing Ramsey Lower Bounds

We will now note how the Ramsey results compare, bringing one of the first order Ramsey lower bounds into the second order context. Recall that we have a translation from the first order context into the second order context. So in fact $\mathsf{Ramsey}(G, n \to (k)_r)$ for $G < 2^{\binom{n}{2}}$, translates roughly to $\mathsf{Ramsey2}(G, n \to (k)_r)$. In fact a number $G < 2^{\binom{n}{2}}$ in the first order context will be translated to a set with bound $\binom{n}{2}$. However we will be working with theories that are robust enough to know this is equivalent to G represented as an adjacency matrix in Ramsey2. We will translate theorem 3.1.2, which says that $\mathrm{S}^1_2 + \mathsf{rWPHP}(\Sigma^b_1)$ proves

$$\exists G < \mathsf{pow}(2, \binom{2^{k/2}}{2}) \; \neg \mathsf{Ramsey}(G, 2^{k/2} \to (k)_2).$$

S^1_2 translates to V^1. The Ramsey statement translates to: $\exists G \; \neg \mathsf{Ramsey2}(G, 2^{k/2} \to (k)_2)$, where $2^{k/2}$ is assumed to exist. For $\mathsf{rWPHP}(\mathtt{R})$, where $\mathtt{R}$ is a two place relation symbol, recall that this was a statement about mapping the numbers $x < 2n$ to the numbers $y < n$. This translates to a principle which maps sets $X < n+1$ to sets $X < n$, which we now define.

Definition 6.4.1 *Let* $\mathtt{R}$ *be a relation symbol with 2 set arguments. Let* $\mathsf{setWPHP}(\mathtt{R})$ *be*

$$\forall X < n+1 \; \exists Y < n \; \mathtt{R}(X,Y) \Rightarrow \exists A \neq B < n+1 \; \exists Y < n \; \mathtt{R}(A,Y) \wedge \mathtt{R}(B,Y).$$

Since Σ_1^b formulae translate to $\Sigma_1^{1,b}$ formula, we obtain the following corollary to theorem 3.1.2.

Corollary 6.4.2 $\mathrm{V}^1 + \mathit{setWPHP}(\Sigma_1^{1,b})$ *proves* $\exists G\ \neg \mathit{Ramsey2}(G, 2^{k/2} \to (k)_2)$.

We can now collect together the series of increasingly stronger Ramsey lower bounds, writing them in a more intuitive manner

1. (Proposition 6.3.2) $\mathrm{V}^0 + \mathsf{PHP}$ proves $R_2(k) > (k-1)^2$.
2. (Theorem 6.3.3) $\mathrm{V}^0 + \mathsf{PHP}$ proves $R_2(k) > \binom{k-1}{3}$.
3. (Theorem 6.3.7) $\mathrm{V}^0 + \mathsf{enum} + \mathsf{modDim} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ proves $R_2(k) > 2^{\left(\frac{\log^2 k}{2^8 \log\log k}\right)}$.
4. (Theorem 3.1.2 and Corollary 6.4.2) $\mathrm{V}^1 + \mathsf{setWPHP}(\Sigma_1^{1,b})$ proves $R_2(k) > 2^{k/2}$.

This shows us very precisely how the increasingly stronger Ramsey claims, with their various proofs, differ from a proof-theoretic point of view.

Chapter 7

Conclusion

As the title of the thesis indicates, we have formalized various aspects of combinatorics in various theories of bounded arithmetic. Two particular methods we looked at were the probabilistic methods and linear algebra methods. The work culminated in applying both of these methods to Ramsey theory. As discussed in the introduction, this work (and that of others in Bounded Arithmetic) can be seen as a beginning for the Reverse Mathematics of finite combinatorics. Such an approach to formalization takes some weak theory as its base theory, adding axioms and proving reversals over this base theory. The base theory should be strong enough to express the basic notions of the area, yet weak enough that it does not blur the distinctions between various theorems of finite combinatorics.

This work suggests taking V^0 as a base theory for finite combinatorics. Using the simple tree objects we have enough expressive power to state many claims. However, we found (in chapter 5) that it was weak enough to allow for distinctions based on what axioms we added to the system. In the standard Reverse Mathematics, only a few extensions of the base theory are needed to capture much of ordinary mathematics. Similarly, we only extend V^0 by a few natural axioms, which suffice to capture much of the finite combinatorics and related tools. Notice that V^1 (or S^1_2), the other natural choice, is too strong for many applications. It would be fine for examining non-constructive proofs which use more than just polynomial time reasoning (e.g. as in the work on the probabilistic methods of chapter 3), but to compare the whole range of theorems from constructive to non-constructive, something weaker like V^0 makes more sense.

Motivated by the issues of Reverse Mathematics, it would be very interesting to prove more reversals. A particularly interesting domain for this would be the probabilistic proofs of chapter 3. Recall that most of these theorems claimed that $S^1_2 + \mathsf{WPHP}$ proved some theorem, say ψ. To prove a reversal over S^1_2 (an appropriate base theory for this context) would mean that we could show $S^1_2 + \psi$ proves WPHP, thus, essentially showing the theorem has no constructive proof. Such

a proof would provide a relative hardness result for ψ, but fall short of an absolute independence result within the hierarchy of S_2 (since we currently do not have an independence result for WPHP when a formulae class is substituted for the relation symbol). However, it should be noted that such issues could still be very difficult, since showing such a reversal would essentially rule out the possibility of finding a simple construction corresponding to the theorem (and often there are simple but tricky constructions corresponding to non-constructive proofs).

Another way to extend the results would be to prove more theorems of the same type within bounded arithmetic. There are many more applications of probability theory and linear algebra methods to combinatorics. Since a number of the proofs have a similar structure, it would be interesting to obtain some more general results which give conditions under which certain kinds of theorems can be proved. For example, since proofs that use the ordinary probabilistic method look very similar, perhaps there is a general theorem that applies to a whole class of theorems proved in this manner.

As a related issue, it would be nice to make the process of formalization more transparent. There are two general ways this could be done. One approach is to stick with the same theory (say V^0 and its extensions), but soup up the "bootstrapping." By this we mean developing more fully structures like simple trees. Another approach would be to work in an extension to type theory, where the structure of the types could allow for a more transparent discussion of the mathematical objects. Such a type theory has been developed for bounded arithmetic by Cook and Urquhart [13]; the system they develop corresponds to an intuitionistic version of S^1_2. It is not currently clear to me how advantageous this change in context is for bounded arithmetic, though I think it would be interesting to at least consider it more seriously. The usefulness of such a transition may be more apparent in comparison to Reverse Mathematics which uses subsystems of second order arithmetic, rather than a higher order logical theory. People have reasons for sticking with the second order framework for that case, but it is not apparent that the same reasons apply to bounded arithmetic.

Appendix A

Pigeonhole Principle Proof

Recall that theorem 4.3.9 stated that

$\mathrm{V}^0 + \mathsf{enum} + \Delta_1^{1,\mathrm{b}}\text{-}\mathsf{CA}$ proves PHP.

Now we prove this, following Woods' proof [47]. Suppose that $\neg\mathsf{PHP}$, so there exists an injection $F : n+1 \hookrightarrow n$ for some F and n. The idea is to define the following sequence of n functions:

$$\begin{aligned} F = \quad & G_0 : && [0,n] \hookrightarrow [0,n-1] \\ & G_1 : && [1,n] \hookrightarrow [0,n-2] \\ & && \vdots \\ & G_{n-1} : && [n-1,n] \hookrightarrow [0,0] \end{aligned}$$

We will describe this sequence of functions by a single set G. We will then show by induction on $i \leq n-1$ (with G as a parameter) that the G_i are injective and have the indicated range. The injection G_{n-1} is a contradiction. G will be a $\Delta_1^{1,b}$ simple tree with structure tree $n\text{—}(n+1)\text{—}n$, thus $G_{\langle i,x\rangle} < n$ will be "$G_i(x)$;" we will write it more informally. Technically we will define $G_i(x) = 0$ for $x < i$, though we really think of the function as having domain $[i,n]$. We define $G_i(x)$ as follows:

- Let $B^{(i,[a,b])}$ be the length $(b-a+1)$ binary vector defined by (for $k < b-a+1$):

$$B_k := \begin{cases} 1 & \text{if } \exists j < i \ F(j) = k+a \\ 0 & \text{otherwise} \end{cases},$$

  i.e. $B^{(i,[a,b])}$ essentially gives the values in the range of F that are in $[a,b]$ and resulted from an input of $< i$, so $\|B^{(i,[a,b])}\| = \|\{j < i \mid F(j) \in [a,b]\}\|$.

- Counting, we set $G_i(x) := \begin{cases} F(x) - \|B^{(i,[0,F(x)])}\| & \text{if } x \geq i \\ 0 & \text{otherwise} \end{cases}$

A key property (proved at the end) is the "monotonicity property:"

$$G_i(u) < G_i(v) \;\Rightarrow\; F(u) < F(v).$$

Using this fact, we can prove the following (we call it $(*)$):

$$G_{i+1}(x) = \begin{cases} G_i(x) & \text{if } G_i(x) < G_i(i) \\ G_i(x) - 1 & \text{if } G_i(x) > G_i(i) \end{cases}$$

When we in fact apply $(*)$, we will know $x > i$ and G_i is injective, so we need not consider the case of $G_i(x) = G_i(i)$. To prove $(*)$:

1. $G_i(x) < G_i(i) \Rightarrow F(x) < F(i) \Rightarrow G_{i+1}(x) = G_i(x)$.
2. $G_i(i) < G_i(x) \Rightarrow F(i) < F(x) \Rightarrow G_{i+1}(x) = G_i(x) - 1$.

In both cases the first implication follows by monotonicity and the second by the definition of G_i and G_{i+1}:

> $G_i(x) = F(x) - \|B^{(i,[0,F(x)])}\|$ and $G_{i+1}(x) = F(x) - \|B^{(i+1,[0,F(x)])}\|$. The only difference between the "B" sets is that the latter one considers $j = i$ and the former does not. If $F(x) < F(i)$, then this has no effect and we get the same B sets, so $G_{i+1}(x) = G_i(x)$. If $F(i) < F(x)$ then the latter B set gets an extra 1, so $G_{i+1}(x) = G_i(x) - 1$.

Now we show by induction on i that G_i is injective on $[i, n]$. Suppose $x \neq y \geq i + 1$ and we want to show $G_{i+1}(x) \neq G_{i+1}(y)$. By inductive hypothesis $G_i(x) \neq G_i(y)$ so assume $G_i(x) < G_i(y)$. The latter inequality along with $(*)$ will be used in the following 3 cases which cover the possible relationship of $G_i(i)$ to $G_i(x)$ and $G_i(y)$; in each case we show $G_{i+1}(x) < G_{i+1}(y)$.

1. If $G_i(y) < G_i(i)$ then $G_{i+1}(y) = G_i(y) > G_i(x) = G_{i+1}(x)$.
2. If $G_i(i) < G_i(x)$ then $G_{i+1}(x) = G_i(x) - 1 < G_i(y) - 1 = G_{i+1}(y)$.
3. If $G_i(x) < G_i(i) < G_i(y)$ then $G_{i+1}(x) = G_i(x) < G_i(y) - 1 = G_{i+1}(y)$.

Since G_i is injective on $[i, n]$ and $x, y > i$, we need not consider the case of $G_i(x) = G_i(i)$ or $G_i(y) = G_i(i)$.

Now we show the range of G_i is $[0, n - i - 1]$ by induction on i. Consider $G_{i+1}(x)$ for $x \geq i + 1$, assuming G_i is injective and has range $[0, n - i - 1]$. Consider two cases ($G_i(x) = G_i(i)$ not possible, as before).

1. If $G_i(x) < G_i(i)$ then $G_{i+1}(x) = G_i(x) \leq G_i(i) - 1 \leq n - i - 1 - 1 = n - (i+1) - 1$.
2. If $G_i(x) > G_i(i)$ then $G_{i+1}(x) = G_i(x) - 1 \leq n - (i+1) - 1$.

Finally, we prove monotonicity:

We show the contrapositive. Assuming $F(u) \geq F(v)$ we show $G_i(u) \geq G_i(v)$. Let $d = F(u) - F(v) \geq 0$. Since $G_i(u) = F(u) - \|B^{(i,[0,F(u)])}\|$ and $G_i(v) = F(v) - \|B^{(i,[0,F(v)])}\|$, to show $G_i(u) \geq G_i(v)$ it suffices to show $\|B^{(i,[0,F(u)])}\| \leq \|B^{(i,[0,F(v)])}\| + d$.

$$\begin{aligned} \|B^{(i,[0,F(u)])}\| &= \|B^{(i,[F(v)+1,F(u)])} \frown B^{(i,[0,F(v)])}\| \\ &= \|B^{(i,[F(v)+1,F(u)])}\| + \|B^{(i,[0,F(v)])}\| \\ &\leq d + \|B^{(i,[0,F(v)])}\|. \end{aligned}$$

That finishes the proof.

In Woods' original context, the pigeonhole principle and counting axioms were schema. Nevertheless we were able to basically formalize that proof making minor adjustments for this context. It could be interesting to derive general connections between schema versions of statements and versions involving only numbers and sets.

Appendix B

Questions

In this appendix we simply gather together all the questions that were posed during the course of this thesis.

1. (Question 3.1.10, Krajíček) Does some theory of bounded arithmetic prove the "Tournament Principle":

$$\mathsf{tournament}(\mathtt{R}, n) \Rightarrow \exists D \subseteq [n]\ (\mathsf{dominating}(\mathtt{R}, D, n) \wedge \mathsf{size}(D) = \log n)?$$

2. (Question 3.1.20) Can the ordinary probabilistic method be formalized in a theory weaker than $\mathrm{S}^1_2 + \mathsf{rWPHP}(\Sigma^b_1)$? For example, does $\mathrm{S}^1_2 + \mathsf{fWPHP}(\Sigma^b_1)$ suffice?

3. (Question 3.2.8) Is $\mathrm{S}^1_2 + \mathsf{fWPHP}(\Sigma^b_1)$ conservative over S^1_2 under certain conditions related to linearity of expectations?

4. (Question 4.3.10) What is the relationship between enum and PHP over V^0?

5. (Question 5.1.3) What stronger special cases of the Oddtown theorem can be proved without linear algebra? Does the proof of theorem 5.1.2 generalize?

6. (Question 5.1.11) To what extent can the reversal of theorem 5.1.6 be extended?

7. (Question 5.2.7) Is there some way for bounded arithmetic to deal with proofs in which one of the steps involves taking the sum of a sequence of rationals?

8. (Question 5.2.14) Can the Non-Uniform Fisher Inequality (the statement in the theorem 5.2.13) be proved in bounded arithmetic?

Bibliography

[1] M. Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the IEEE 29th Annual Symposium on Foundations of Computer Science*, pages 346 – 355, 1988.

[2] N. Alon. A parallel algorithmic version of the local lemma. *Random Structures and Algorithms*, 2(4):367–378, 1991.

[3] N. Alon, L. Babai, and H. Suzuki. Multilinear polynomials and Frankl-Ray-Chaudhuri-Wilson type intersection theorems. *Journal of Combin. Th. A*, 58:165–180, 1991.

[4] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method*. John Wiley and Sons, New York, 1992.

[5] L. Babai and P. Frankl. Linear algebra methods in combinatorics (with applications to geometry and computer science), preliminary version 2. 1992.

[6] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák, and A. Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 200 – 221. ACM Press, 1992.

[7] J. Beck. An algorithmic approach to the Lovász local lemma. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[8] E. Berlekamp. On subsets with intersections of even cardinality. *Canadian Math. Bull.*, 12:363–366, 1969.

[9] Samual Buss. *Bounded Arithmetic*. Bibliopolis, Italy, 1986.

[10] M. Chiari and J. Krajíček. Lifting independence results in bounded arithmetic. *Archive for Mathematical Logic*, 38(2):123–138, 1999.

[11] S. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the 7th Annual ACM Symposium on Theory of Computing*. ACM Press, 1975.

[12] S. Cook. Csc 2429s notes. course notes (on web site), 2002.

[13] S. Cook and A. Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals os Pure and Applied Logic*, 63:103–200, 1993.

[14] P. D'Aquino. Local behavior of the Chebyshev theorem in models of $I\Delta_0$. *Journal of Symbolic Logic*, 57:12–27, 1992.

[15] P. D'Aquino. Pell equations and exponentiation in fragments of arithmetic. *Annals of Pure and Applied Logic*, 77:1–34, 1996.

[16] P. D'Aquino. Solving pell equations locally in models of $I\Delta_0$. *Journal of Symbolic Logic*, 63:402–410, 1998.

[17] P. D'Aquino and A. Macintyre. Non-standard finite fields over $I\Delta_0 + \Omega_1$. *Israel Journal of Mathematics*, 117:311–333, 2000.

[18] N. de Bruijn and P. Erdős. On a combinatorial problem. *Indagationes Math.*, 10:421–423, 1948.

[19] M. Deza, P. Frankl, and N. Singhi. On functions of strenth t. *Combinatorica*, 3:331–339, 1983.

[20] C. Dimitracopoulos. *Matijasevic's theorem and fragments of arithmetic*. PhD thesis, University of Manchester, 1980.

[21] P. Erdős. Some remarks on the theory of graphs. *Bulletin of the American Mathematics Society*, 53:292–294, 1947.

[22] P. Erdős. On a combinatorial problem I. *Nordisk Mat. Tidskrift*, pages 5–10, 1963.

[23] P. Erdős. On a problem in graph theory. *Mathematical Gazette*, 1963.

[24] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal et al., editor, *Infinite and Finite Sets*, pages 609–628. North-Holland, 1975.

[25] P. Frankl and R.M. Wilson. Intersection theorems with geometric consequences. *Combinatorica*, 1(4):357–368, 1981.

[26] R. Graham, B. Rothschild, and J. Spencer. *Ramsey Theory*. John Wiley and Sons, New Yok, 2nd edition, 1990.

[27] P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1998.

[28] J. Håstad. *Computational Limitations of Small-Depth Circuits*. ACM Doctoral Dissertation Series. MIT Press, Cambridge, Mass., 1987.

[29] E. Jerabek. Dual weak pigeonhole principle, boolean complexity, and derandomization. submitted, 2003.

[30] R. Kaye. *Models of Peano Arithmetic*, volume 15 of *Oxford Logic Guides*. Clarendon Press, New York, 1991.

[31] J. Krajíček. No counter-example interpretation and interactive computation. In Y.N. Moschovakis, editor, *Logic From Computer Science*, volume 21, pages 287–293. Springer-Verlag, 1989.

[32] J. Krajíček. Exponentiation and second order bounded arithmetic. *Annals of Pure and Applied Logic*, 48:261–276, 1990.

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

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

[35] A. Maciel, T. Pitassi, and A. Woods. A new proof of the weak pigeonhole principle. submitted, 2000.

[36] K. Majumdar. On some theorems in combinatorics relating to incomplete block designs. *Ann. Math. Stat.*, 24:377–389, 1953.

[37] Z. Nagy. A certain constructive estimate of the Ramsey number (Hungarian). *Matematikai Lapok*, 23:301–302, 1972.

[38] R. Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36:494–508, 1971.

[39] J. Paris, A. Wilkie, and A. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *The Journal of Symbolic Logic*, 53(4):1235–1244, 1988.

[40] P. Pudlák. Ramsey's theorem in bounded arithmetic. In E. Borger, editor, *Lecture Notes in Computer Science*, volume 533, pages 308–312. Springer-Verlag, 1991.

[41] A. Razborov. An equivalence between second order bounded domain bounded arithmetic and first order bounded arithmetic. In P. Clote and J. Krajíček, editors, *Proof Theory and Computational Complexity*, pages 247–277. Oxford University Press, 1993.

[42] S. Simpson. *Subsystems of Second Order Arithmetic*. Springer-Verlag, New York, 1999.

[43] M. Soltys and S. Cook. The proof complexity of linear algebra. submitted, 2003.

[44] G. Takeuti. S^i_3 and $V^i_2(\mathsf{BD})$. *Archive for Mathematical Logic*, 29:149–169, 1990.

[45] G. Takeuti. RSUV isomorphism. In P. Clote and J. Krajíček, editors, *Proof Theory and Computational Complexity*, pages 364–386. Oxford University Press, 1993.

[46] N. Thapen. A model-theoretic characterization of the weak pigeonhole principle. *Annals of Pure and Applied Logic*, 118:175–195, 2002.

[47] A. Woods. *Some problems in logic and number theory and their connections*. PhD thesis, University of Manchester, 1981.

[48] D. Zambella. Notes on polynomially bounded arithmetic. *Journal of Symbolic Logic*, 61(3):942–966, 1996.