universität
wien

MASTERARBEIT/MASTER'S THESIS

Titel der Masterarbeit/Title of the Master's Thesis

"Jan Krajíček's Forcing Construction and Pseudo Proof Systems"

verfasst von/submitted by

Jan Felix Maly, B.Sc.

angestrebter akademischer Grad/in partial fulfillment of the requirements for the degree of

Master of Science (M.Sc.)

Wien, 2016/Vienna 2016

Studienkennzahl lt. Studienblatt/ degree programme code as it appears on the student record sheet:	A 066821
Studienrichtung lt. Studienblatt/ degree programme as it appears on the student record sheet:	Masterstudium Mathematik UG2002
Betreut von/Supervisor:	Dr. Moritz Müller, Privatdoz.

Contents

Introduction

This thesis is intended to give an introduction to the method of forcing with random variables introduced by Jan Krajíček in his book ‘Forcing with random variables and proof complexity’ in 2010 ([33]). A book that, in the words of Sam Buss, “ gives a fundamentally new approach to model-theoretic forcing, as well as to independence results in bounded arithmetic and proof complexity” [11]. However it is also “research-level exposition of new topics that have not appeared in the literature” [11]. Therefore it may not be accessible to readers without the necessary background in bounded arithmetic and proof complexity. This text aims to make the general idea of the method accessible to every reader with a basic understanding of logic and complexity theory.

The first chapter is an introduction to topics needed to understand the forcing method that exceed what is usually covered in beginners courses on logic and theoretical computer science. The second chapter of this text introduces the forcing method in its general form, covering chapter 1 and 2 as well as large parts of chapter 3 of [33]. The third chapter treats an application of the forcing method, presented in section 24.1 of [33]. Finally, Chapter four covers some new results about pseudo proof systems, a topic proposed in chapter 24.4 of [33], closely related to the forcing method.

Prerequisites

This text is supposed to be an introduction, therefore, wherever possible, all necessary concepts are introduced. However, it doesn’t seem practical to keep the text entirely self-contained. Hence, it is assumed that the reader has some knowledge in logic and complexity theory (as well as some very basic knowledge in algebra). Any (undergraduate) course in logic and theoretical computer science will be more than sufficient. Alternatively, introductory textbooks as Herbert Endertons “A mathematical introduction to logic” [23] and “Computational complexity: a modern approach” by Sanjeev Arora and Boaz Barak [5] cover everything that appears in this text and much more.

Concretely, familiarity with the following items is expected:

- It is assumed that the reader is familiar with propositional logic, i.e he knows how propositional formulas are build up from variables and the logical symbols (in this text we will use $\{\neg, \wedge, \vee, \top, \bot\}$ as our primitive logical symbols). He knows how to evaluate a formula under a given assignment and is familiar with truth tables.

- Furthermore, it is assumed that the reader is familiar with first order logic ($\exists$ and $\forall$ are used as additional primitive logic symbols). He knows how first order languages L are defined (observe that we will treat constants as 0-arry functions). Moreover he is acquainted with at least one proof system, knows how L-models are defined and is aware of the satisfaction relation $\models$. Ideally he has also seen a proof of the completeness theorem.

- Additionally, it is assumed that the reader is familiar with at least one possible definition of a Turing machine, is aware of the complexity class P and able to deal with polynomial time functions confidently. Ideally he should also know about the class NP, this is, however, not mandatory.

The items above should suffice to follow all technical parts, i.e definitions, propositions, theorems etc.

Bounded arithmetic and proof complexity

Krajíček developed the forcing method described in this thesis as a tool to build models of bounded arithmetic. The term bounded arithmetic denominates not a single theory but a collection of arithmetical theories whose principal axiom schema is a form of induction for predicates with limited computational complexity. The first theories of bounded arithmetic considered in the literature were $I\Delta_0$ introduced by Parikh ([38]) and PV introduced by Stephen Cook in 1975 [17]. The systematic study of theories of bounded arithmetic was started by Samuel Buss in his PhD thesis from 1985 [12], which introduced the two interleaved hierarchies S_2^i and T_2^i. The study of bounded arithmetic is very closely linked to the study of proof complexity, which asks about the length of proofs of propositional formulas in different proof systems. One can even think of proof systems as non-uniform versions of these theories in a precise technical sense ([33]). Furthermore, both topics are closely linked to fundamental open questions in complexity theory.

Proving superpolynomial lower bounds for (all abstract) proof systems is a possible approach to establish $NP \neq coNP$ and consequently $P \neq NP$. This approach is sometimes called Cook's Program. Unfortunately, the current state of art in proof complexity is still very far away from solving this problem (see [13]). Proving independence results for theories of bounded arithmetic is very much related to proving such lower bounds. Hence a strong forcing method in bounded arithmetic could be a valuable tool for proof complexity.

All this notwithstanding, this text will not treat any theory of bounded arithmetic in detail. In set theory, forcing is nearly always used to build a model of $ZFC(+X)$ and the proof that a forcing extension models ZFC proceeds by proving every axiom of ZFC individually. The situation is completely different for Krajíček's forcing in bounded arithmetic. On the one hand, there are many theories of bounded arithmetic and the forcing framework allows to build models for all of them. The disadvantage of this approach is that it is not possible to give a strong general statement about which theories are modeled by forcing extensions. Therefore this has to be checked in every application individually. On the other hand, however, it is possible in many applications to use known properties of the respective theory to avoid checking all axioms by hand. Consequently, an introduction to the forcing formalism has no benefit from treating bounded arithmetic in detail. The reader should, however, be aware that bounded arithmetic and proof complexity provide the context and are the targeted area of application for the method described in this thesis. Therefore it might be beneficial to have a look at Krajíček's book [32] for a thorough treatment of proof complexity and bounded arithmetic.

Chapter 1

Theoretical background

In this chapter we will provide an introduction to the theoretical background that is necessary, in addition to the topics designated as prerequisites in the introduction, to understand the presentation of the forcing method in chapter 2. These introductions are very concise and target-oriented towards the forcing method. For more comprehensive discussions of these topics the reader has to consult the literature proposed in the text.

1.1 Model theory of arithmetic

This section will introduce some model theoretic results with a special focus on models of a very strong theory of arithmetic. To define this theory, we have to fix a language first.

Definition 1.1. Let $\mathbf{L_{all}}$ denote the (uncountable) language that consists of symbols for all possible functions and relations on the natural numbers. In particular L_{all} contains constants (which are viewed as 0-ary functions) for all $n \in \mathbb{N}$.

There is one obvious L_{all}-model.

Definition 1.2. The **standard model for $\mathbf{L_{all}}$** has the set of natural numbers as universe and interprets every function symbol as intended. That means the symbol $f \in L_{all}$ for a function $f^{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ is interpreted in the standard model by $f^{\mathbb{N}}$. Relation symbols are treated analogously. Abusing notation a bit we will just write $\mathbb{N}$ for the standard model.

We can use this model to define the "true" first order theory of arithmetic.

Definition 1.3. Let $\mathbf{Th_{L_{all}}}(\mathbb{N})$ denote the collection of all L_{all}-sentences true in the standard model $\mathbb{N}$. We call this theory **true arithmetic**.

In order to talk about models of this theory, other then the standard model, we have to recall a few standard notions from model theory. More details can be found for example in [14].

Definition 1.4. Let M be a model in a language L. For any subset $A \subseteq M$ we define $\mathbf{L_A} := \{c_a \mid a \in A\} \cup L$ to be the language obtained from L by adding a new constant for every element of A. Then, let M_A be the L_A-model that coincides with M on L and interprets c_a as a. A **type** over A is a set $p(\overline{x})$ of L_A-formulas with free variables in $\overline{x}$ such that for any finite subset $p_0(\overline{x}) \subseteq p(\overline{x})$ there is a tuple $\overline{b} \in M^{<\omega}$ such that $M_A \models \phi(\overline{b})$ holds for all $\phi(\overline{x}) \in p_0(\overline{x})$.

A type $p(\overline{x})$ is **complete** if for every L_A-formula $\phi(\overline{x})$ with free variables in $\overline{x}$ either $\phi(\overline{x}) \in p(\overline{x})$ or $\neg\phi(\overline{x}) \in p(\overline{x})$ holds. We say a type $p(\overline{x})$ is **realized** in M if there is a $\overline{b} \in M^{<\omega}$ such that $M_A \models \phi(\overline{b})$ holds for all $\phi(\overline{x}) \in p(\overline{x})$. We say a model M is **κ-saturated** if it realizes all complete (and therefore all[1]) types over all $A \subseteq M$ with cardinality less than κ.

It is not clear, a priori, that $\aleph_1$-saturated models of $Th_{L_{all}}(\mathbb{N})$ exist. The standard model, for example, is not $\aleph_1$ saturated.

Example 1.1. The standard model $\mathbb{N}$ is not $\aleph_1$-saturated. Take $A = \mathbb{N}$. Then A is trivially countable[2]. We define the formula $\phi_n(x) := \neg(x = n)$ for all $n \in \mathbb{N}$ and $p(x) := \{\phi_n \mid n \in \mathbb{N}\}$. $p(x)$ is a type as for any finite collection $\{\phi_{n_1}, \dots \alpha_{n_k}\}$ there is some n^* such that $n^* \neq n_i$ holds for all $i \in \{1, 2 \dots k\}$. But this type can not be realized in $\mathbb{N}$ as for every $n \in \mathbb{N}$ we have $\mathbb{N} \not\models \alpha_n(n)$. This argument shows even more generally that no countable (infinite) model can be $\aleph_1$-saturated.

However, there are plenty saturated models available. To prove this we have to dig a bit deeper into model theory. First we recall two of the most important results in model theory and logic in general.

Reminder 1.1. The two basic theorems that build the foundation of modern model theory are the completeness and compactness theorem. The first one, proven by Kurt Gödel in 1929 [26], can be formulated as the assertion that every consistent first order theory has a model. The latter, a direct consequence of the completeness theorem, states that a set of first order sentences has a model if and only if every finite subset of it has a model.

[1] It is easy to see that every type can be completed the same way every theory can be completed.

[2] As we have constants in L_{all} for all natural numbers anyway we could also choose $A = \emptyset$. The argument as presented has the advantage that it works also for weaker languages

Proofs and further discussion of these theorems can be found in any beginners textbook on logic (e.g. [23]).

Additionally to these two basic facts we will need a few slightly more advanced concepts about models. For a more thorough treatment of these see for example [14].

Definition 1.5. Let M be a model, $N \subseteq M$ a submodel of M. We say M is an **elementary extension** of N, in symbols $N \preceq M$ if for every formula $\phi(\overline{x})$ and every tupel $\overline{a}$ of elements of N we have

$$M \models \phi(\overline{a}) \Leftrightarrow N \models \phi(\overline{a})$$

Definition 1.6. Let M be a model of a language L. We write $L(M)$ for the language obtained by adding to L a new constant c_a for all $a \in M$. Let M^* be the extension of M to $L(M)$ that interprets c_a by a for all a in M. We call the set of all $L(M)$-sentences true in M^* the **elementary diagram** of M.

If we have two models $N \subseteq M$ it is easy to see that M is an elementary extension of N if and only if the $L(N)$ extension of M interpreting c_a by a for all $a \in N$ is a model of the elementary diagram of N. See for example [14]. This allows us to find elementary extensions that realize all types over a model.

Lemma 1.1. *Let M be a model in a language L. There exists an elementary extension N of M such that all types over M are realized in N.*

Observe that $M \preceq N$ implies $M \subseteq N$ therefore it makes sense to ask if N realizes types over M.

Proof. We start with the language $L(M)$. For every type $p(x)$ over M we introduce a new constant c_p to generate the language

$$L^* := L(M) \cup \{c_p \mid \text{p is a type over } M\}$$

Now let T be the L^* theory consisting of the elementary diagram of M and, for all types p over M, the sentences $\phi(c_p)$ for all $\phi(x) \in p(x)$. Observe that every finite subset T_0 of T has a model because M can be extended to a model of T_0 by definition of type and elementary diagram. Therefore, by the compactness theorem, there is a model N that models T. As N models the elementary diagram of M it is easy to see that there is a isomorphic model N' such that $M \subseteq N'$ holds. The restriction of N' to L is an elementary extension of M and it realizes all types $p(x)$ via the N'-interpretation of c_p. □

This lemma does not provide a κ-saturated model because, in general, there are new subsets in N and, therefore, also new types. To solve this problem, we iterated the lemma transfinitely often.

Theorem 1.2. *Let T be a consistent theory. Then, for every cardinal κ, there exists a κ-saturated model of T.*

To make the proof a bit more readable we will only prove the special case that is relevant to us. The proof idea stays the same for the general case, a proof of which can be found in [14].

Theorem 1.3. *A $\aleph_1$-saturated model of true arithmetic exists.*

Proof. We construct an elementary chain of models N_ξ of length $\aleph_1$ such that for each ξ the model N_ξ is a elementary extension of the standard model $\mathbb{N}$. We start with $N_0 = \mathbb{N}$. In each limit step ν we set $N_\nu = \bigcup_{\xi<\nu} N_\xi$. For each successor step $\xi + 1$ we let $N_{\xi+1}$ be an elementary extension of N_ξ that realizes all types over N_ξ. Such a model exists by lemma 1.1. Finally we set $N := N_{\aleph_1} = \bigcup_{\xi<\aleph_1} N_\xi$. Clearly for every successor step $\xi + 1$, if $\mathbb{N} \preceq N_\xi$ holds so does $\mathbb{N} \preceq N_{\xi+1}$ because $\preceq$ is transitive. Furthermore, it is easy to see that $N_{\xi'} \preceq \bigcup_{\xi<\nu} N_\xi$ holds for $\xi' < \nu$. Therefore, by transfinite induction $\mathbb{N} \preceq \mathbb{N} = N_0$ implies $\mathbb{N} \preceq N$. But every elementary extension of the standard model is, by definition, a model of true arithmetic, hence N is a model of true arithmetic.

Furthermore, N is $\aleph_1$-saturated. Assume otherwise that there is some countable $X \subseteq N$ and a type $p(x)$ over X such that N does not realize $p(x)$. For every $a \in X$ let $\mathrm{rank}(a)$ be the minimal ξ such that a is an element of N_ξ. Then $\xi^* := \max(\{\mathrm{rank}(a) \mid a \in X\})$ is smaller than $\aleph_1$ because $\aleph_0$ is not cofinal[3] in $\aleph_1$. But then we know that there is an element b in N_{ξ^*+1} such that N_{ξ^*+1} realizes p trough b. By the argument above we know $N_{\xi^*+1} \prec N$ and by definition we know that b is an element of N. This means N realizes p through b. Contradiction. □

We need a few model theoretic results regarding nonstandard models. First of all it is easy to see that every model of true arithmetic[4] contains an isomorphic copy of $\mathbb{N}$.

Example 1.2. There are constants c_n for every $n \in \mathbb{N}$ in L_{all}, hence, $Th_{L_{all}}$ is the elementary diagram of $\mathbb{N}$. Therefore, every model of true arithmetic contains a submodel N isomorphic to $\mathbb{N}$. We call N the **standard part** of

[3] This is basic concept of set theory. We won't discuss cofinallity in this text. If the reader is not familiar with this concept he can consult [35].

[4] Actually this is true even for most weaker systems of arithmetic.

M. Accordingly elements of N are called **standard numbers** and elements of $M \setminus N$ are called **nonstandard numbers**. Observe that a nonstandard number m has to be bigger in M than all standard numbers because in $\mathbb{N}$, hence also in M, it is true that a number smaller than $k \in \mathbb{N}$ must be equal to a constant from the set $\{c_l \in \mathbb{N} \mid l < k\}$. From now on we will identify this isomorphic copy N with $\mathbb{N}$ to make our presentation more readable and treat $\mathbb{N}$ as a (proper) subset of M for a nonstandard model of true arithmetic M.

We will finish this section by proving three important properties of ($\aleph_1$-saturated) nonstandard models of true arithmetic. We will present a variation of the more general proofs from [29]. The first proposition describes the so called overspill, a general property of nonstandard models of arithmetic that follows easily from induction but has many important applications in the model theory of arithmetic.

Proposition 1.4 (Overspill)**.** *Let M be a nonstandard model of true arithmetic and let $\phi(x)$ be a L_{all}-formula, possibly with parameters from M. If $M \models \phi(n)$ holds for all $n \in \mathbb{N}$, then there is also a nonstandard $m \in M \setminus \mathbb{N}$ such that $M \models \phi(m)$ holds.*

Proof. We want to prove this by contradiction. Assume there is a L_{all} formula $\phi(x)$ that holds in M for all $n \in \mathbb{N}$ but for no $m \in M \setminus \mathbb{N}$. We know that $\mathbb{N}$ models induction for all formulas $\psi(x)$. That means

$$\mathbb{N} \models (\psi(0) \land \forall x(\psi(x) \to \psi(x+1))) \to \forall y \psi(y)$$

Hence, this is also true for ϕ. Because M is a model of true arithmetic this carries over to M so we know

$$M \models (\phi(0) \land \forall x(\phi(x) \to \phi(x+1))) \to \forall y \phi(y) \ (1)$$

As 0 is a element of $\mathbb{N}$ we know $M \models \phi(0)$ by assumption. Moreover,the successor of every $k \in \mathbb{N}$ is also in $\mathbb{N}$ so we know $M \models \forall k(\phi(k) \to \phi(k+1))$, too. But then (1) tells us that $M \models \forall x \phi(x)$ must be true. Hence, $\phi(m)$ holds for all nonstandard $m \in M$. This contradicts our assumption!

□

The next two statements are consequences of the $\aleph_1$-saturation. Basically, these two results are the reason we need the saturation in the first place. Before we can state the first proposition we need to recall a well known fact.

Reminder 1.2. Sets and sequences of elements of model are in general not themselves elements of the model. If we have a model of arithmetic we can bypass this problem somewhat because we can use codes to talk about finite sequences and sets. This idea dates back to Kurt Gödel. Some treatment of coding in a general setting can be found in [23].

We don't need very sophisticated methods of coding because we want to work in an L_{all} model M. For example we could code a finite sequence of natural numbers $k_1, k_2 \dots k_l$ using the first l primes $p_1, p_2 \dots p_l$ by the number $p_1^{k_1+1} p_2^{k_2+1} \dots p_l^{k_l+1}$. Then, obviously, every sequence has a unique code c and we can read the length and the individual elements of the sequence of the code using L_{all} functions *leng* and *elem*, such that $leng(c) = l$ and $elem(c, i) = k_i$ holds for every sequence $(k_i)_{i<l}$ and its code c. If $(s_i)_{i<l}$ is a sequence with $s_i, l \in M$ we will write in the following $s \in M$ to indicate that s has a code in M, i.e there is a element $m \in M$ such that $leng(m) = l$ and $elem(m, i) = s_i$ for all $i < l$ holds in N.

Proposition 1.5. *Let M be a $\aleph_1$-saturated nonstandard model of true arithmetic and let $(a_k)_{k\in\mathbb{N}}$ be a countable sequence of elements of M. Then there exists a sequence $(b_i)_{k<m}$ of nonstandard length $m \in M \setminus \mathbb{N}$ that is coded in M and satisfies $a_k = b_k$ for all $k \in \mathbb{N}$.*

Proof. Consider the set of formulas $p(x)$ using parameters from $A := \{a_k \in M \mid k \in \mathbb{N}\}$ that contains for every $j \in \mathbb{N}$ a formula $\alpha_j(x)$ expressing: 'There is a $m \in M$ such that x codes a sequence $(b_i)_{i<m}$ of length $m \geq j$ and $a_k = b_k$ for all $k < j$.' To see that the $p(x)$ is a type, look at any finite collection $p_0(x)$ of formulas $\alpha_j(x)$. There is a maximal index j_0 which means $\alpha_j \in p_0$ implies $j \leq j_0$. We can choose m to be the code of $(a_k)_{k<j_0}$. Then $\alpha_{j_0}(m)$ is true in M by definition. Furthermore, this implies $\alpha_j(m)$ for all $j \leq j_0$, hence all formulas in $p_0(m)$ are true in M. Therefore, $p(x)$ is a type in M over A. Finally A is the union of two countable sets and therefore countable, so we can use the $\aleph_1$-saturation to find some element $m^* \in M$ that realizes (a completion of) p. This m^* codes a sequence $(b_i)_{i<m}$ of length m. Obviously m is nonstandard because $m \geq j$ holds for every $j \in \mathbb{N}$. Furthermore, we get $b_k = a_k$ for every $k \in \mathbb{N}$ because there is a $j > k$ such that $\alpha_j(m^*)$ holds in M which implies the equality.

□

Proposition 1.6. *Let M be a $\aleph_1$-saturated nonstandard model of true arithmetic. If $(A_k)_{k\in\mathbb{N}}$ is a countable sequence of definable[5] subsets of M such that $\bigcap_{i<k} A_i \neq \emptyset$ holds for all $k \geq 1$ we know $\bigcap_{k\in\mathbb{N}} A_k \neq \emptyset$.*

[5] In this text, definable always means definable with parameters if not stated otherwise

Proof. Let $\alpha_k(x)$ be the defining formula for A_k and P_k the (finite) set of parameters used in α_k. Then $p(x) := \{\alpha_k \mid k \in \mathbb{N}\}$ is a type over $P := \bigcup_{k \in \mathbb{N}} P_k$ because every finite subset $p_0(x)$ of $p(x)$ has the form $p_0(x) = \{\alpha_{k_1}(x), \alpha_{k_2(x)} \dots \alpha_{k_l}(x)\}$. Therefore, $p_0(x)$ is realized by any element of $\bigcap_{i<k^*} A_i$ where $k^* = \max\{k_1, k_2 \dots k_l\}$. By assumption such an element exists. Furthermore, P is obviously countable so we can use $\aleph_1$-saturation to find a $m \in M$ such that $M \models p(m)$. But this means $M \models \alpha_k(m)$ for every $k \in \mathbb{N}$ which implies $m \in A_k$ for all $k \in \mathbb{N}$. This implies $m \in \bigcap_{k \in \mathbb{N}} A_k$ hence the intersection is not empty. □

It is worth noting that $\bigcap_{k \in \mathbb{N}} A_k$ need not be definable, as the next example shows.

Example 1.3. Consider the definable sets $A_k := \{n \mid n > k\}$ for all $k \in \mathbb{N}$ Their intersection is empty in $\mathbb{N}$. In a nonstandard model M on the other hand their intersection equals $M \setminus \mathbb{N}$. But proposition 1.4 tells us that this set can not be definable in M as $n \notin M \setminus \mathbb{N}$ is true if and only if n is a natural.

1.2 Boolean algebras

In this section we will quickly introduce all relevant concepts regarding boolean algebras. Boolean algebras are named after George Boole, an English mathematician and pioneer of logic, who introduced an algebraic structure similar to the modern notion of a boolean algebra in [10] in 1847. For more information about George Boole and his contributions to logic see [28]. Today boolean algebras are studied mainly in set theory as well as in algebra, see, for example, [35].

Definition 1.7. A **boolean algebra** $\mathcal{A}$ consists of a set A, together with two binary functions $\wedge$ and $\vee$, a unary function $\neg$ and two constants $0_{\mathcal{A}}$ and $1_{\mathcal{A}}$. Furthermore, A has to be closed under these functions and for all elements a, b and c of A, the following axioms must hold:

$$\begin{aligned} a \vee (b \vee c) &= (a \vee b) \vee c & a \wedge (b \wedge c) &= (a \wedge b) \wedge c \\ a \vee b &= b \vee a & a \wedge b &= b \wedge a \\ a \vee 0_{\mathcal{A}} &= a & a \wedge 1_{\mathcal{A}} &= a \\ a \vee (b \wedge c) &= (a \vee b) \wedge (a \vee c) & a \wedge (b \vee c) &= (a \wedge b) \vee (a \wedge c) \\ a \vee \neg a &= 1_{\mathcal{A}} & a \wedge \neg a &= 0_{\mathcal{A}} \end{aligned}$$

We can use the function $\wedge$ to define a natural partial order on any boolean algebra.

Definition 1.8. Let $\mathcal{A}$ be a boolean algebra. We define the **relation** $\leq$ on A for $a, b \in A$ by $a \leq b$ if $a \wedge b = a$.

This relation is a partial order as intended. The proof is a simple application of the axioms of a boolean algebra.

Lemma 1.7. *The relation $\leq$ on a boolean algebra $\mathcal{A}$ as defined above is a partial order with maximal element $1_{\mathcal{A}}$ and minimal element $0_{\mathcal{A}}$.*

Proof. We have to prove that $\leq$ is reflexive, antisymmetric and transitive.

- $\leq$ is reflexive: $a \wedge a = (a \wedge a) \vee 0_{\mathcal{A}} = (a \wedge a) \vee (a \wedge \neg a) = a \wedge (a \vee \neg a) = a \wedge 1_{\mathcal{A}} = a$ by the axioms of a boolean algebra.
- $\leq$ is antisymmetric: Assume $a \wedge b = a$ and $b \wedge a = b$. Then $a = a \wedge b = b \wedge a = b$ hence $a = b$.
- $\leq$ is transitive: Assume $a \wedge b = a$ and $b \wedge c = b$ then $a = a \wedge b = a \wedge (b \wedge c) = (a \wedge b) \wedge c = a \wedge c$.
- Maximal and minimal element: $a \wedge 1_{\mathcal{A}} = a$ and $a \wedge 0_{\mathcal{A}} = 0_{\mathcal{A}}$ hold for every $a \in \mathbb{A}$ by definition.

□

The following lemma will be useful later on.

Lemma 1.8. *Let $\mathcal{A}$ be a boolean algebra and $\leq$ the partial order defined above. For all $a, b, c \in \mathcal{A}$ the following holds:*

- *$a \leq b$ implies $\neg b \leq \neg a$.*
- *$a \leq b$ implies $a \vee c \leq b \vee c$.*

Proof.

- Assume $a \wedge b = a$, then the following holds: $\neg b = \neg b \wedge (\neg a \vee a) = (\neg b \wedge \neg a) \vee (\neg b \wedge a) = (\neg b \wedge \neg a) \vee (\neg b \wedge (a \wedge b)) = (\neg b \wedge \neg a) \vee (a \wedge 0_{\mathcal{A}}) = \neg b \wedge \neg a$.
- Assume $a \wedge b = a$, then the following holds: $(a \vee c) \wedge (b \vee c) = (a \wedge b) \vee c = a \vee c$.

□

Using this partial order we can identify additional useful properties that a boolean algebra can enjoy. These properties can be viewed as nice closure properties of the underlying set.

Definition 1.9. We say a boolean algebra $\mathcal{A}$ is a **σ-algebra** if every countable set $X \subseteq A$ has a supremum with respect to $\leq$ in A. Furthermore, a boolean algebra $\mathcal{A}$ is called **complete** if every set $X \subseteq A$ has a supremum with respect to $\leq$ in A. We write $\bigvee_{i \in I} a_i$ for the supremum of the family $\{a_i\}_{i \in I}$ with $a_i \in A$ and $\bigwedge_{i \in I} a_i$ for the infimum of the same family.

Finally using the partial order we can define antichains and the ccc-property for boolean algebras, two concepts of uttermost importance in set theoretic forcing (see [35]). In order to simplify a few proofs later on we will use strong antichains instead of ordinary (weak) antichains. As only strong antichains appear in this text we will omit the word 'strong'.

Definition 1.10. Let $\mathcal{A}$ be a boolean algebra. We call two elements $a, b \in A$ **disjoint** if $a \wedge b = 0_{\mathcal{A}}$. A subset X of A is called an **antichain** if all elements of X are mutually disjoint. An antichain X is **maximal in** $B \subseteq A$ if $b \neq 0_{\mathcal{A}}$ and $b \in B \setminus X$ together imply that there is a $x \in X$ such that $b \wedge x \neq 0_{\mathcal{A}}$ holds. Finally we say $\mathcal{A}$ has the **ccc-property** if every antichain in A has only countably many elements.

1.3 Boolean valued models

Boolean valued models were first described in [45] to give an alternative interpretation of Cohens then newly introduced forcing method [16]. More information about boolean valued models can be found, for example, in [31]. Sentences aren't just true or false in these models, but rather take truth values from a boolean algebra. This is a generalization of the concept of a model, as the truth values 0 and 1 together with the logical connectives in the traditional case also form a boolean algebra. A boolean valued model is defined as follows.

Definition 1.11. Let $\mathcal{A}$ be a complete boolean algebra and let L be a language. Then a **boolean valued L-model $\mathcal{M}_\mathcal{A}$ over $\mathcal{A}$** consists of a set $M_\mathcal{A}$, called the universe of $\mathcal{M}_\mathcal{A}$, together with for all $k \in \mathbb{N}$ a function $f_{\mathcal{M}_\mathcal{A}} : M_\mathcal{A}^k \to M_\mathcal{A}$ for every k-ary function symbol f in L. Finally $\mathcal{M}_\mathcal{A}$ contains for every relation symbol $R \in L$ and every k-tuple $(m_1, m_2 \dots m_k) \in M_\mathcal{A}^k$, where k is the arity of R, a **boolean valuation** $[\![R(m_1, m_2, \dots m_k)]\!] \in A$, as well as a **boolean valuation** $[\![m_1 = m_2]\!] \in A$ for all pairs $(m_1, m_2) \in M_\mathcal{A}$.

Terms are evaluated as in classical models and for every L-sentence with parameters from $M_\mathcal{A}$ the **truth value $[\![\phi]\!]$ of ϕ in $\mathcal{M}_\mathcal{A}$** is defined by the following induction on formula complexity:

- For $\phi = R(t_1, t_2 \dots t_k)$ where R is a relation and $t_1, t_2, \dots t_k$ are terms with evaluation $m_1, m_2, \dots m_k \in M_\mathcal{B}$ the truth value $[\![\phi]\!]$ is given by the boolean valuation $[\![R(m_1, m_2, \dots m_k)]\!]$.
- For $\phi = (t_1 = t_2)$ where t_1 and t_2 are terms with evaluation $m_1 \in M_\mathcal{A}$ and $m_2 \in M_\mathcal{A}$ respectively the truth value $[\![\phi]\!]$ is given by the boolean valuation $[\![m_1 = m_2]\!]$.
- $[\![\phi \wedge \psi]\!] := [\![\phi]\!] \wedge [\![\psi]\!]$, $[\![\phi \vee \psi]\!] := [\![\phi]\!] \vee [\![\psi]\!]$ and $[\![\neg\phi]\!] := \neg[\![\phi]\!]$
- $[\![\exists x \phi(x)]\!] := \bigvee\limits_{m \in M_\mathcal{A}} [\![\phi(m)]\!]$
- $[\![\forall x \phi(x)]\!] := \bigwedge\limits_{m \in M_\mathcal{A}} [\![\phi(m)]\!]$

We say a sentence ϕ is **$\mathcal{M}_\mathcal{A}$-valid** if its truth value $[\![\phi]\!]$ is $1_\mathcal{A}$.

Its worth noting that the $\forall$ and the $\exists$ step in the inductive definition are only well defined because we require $\mathcal{A}$ to be complete.

Boolean valued models have many nice properties. We prove only two properties we intend to use later on.

Notation 1.1. As usual, we write in the following $\phi \to \psi$ as a shorthand for $\neg\phi \vee \psi$ and $\phi \leftrightarrow \psi$ as a shorthand for $(\phi \to \psi) \wedge (\psi \to \phi)$.

Proposition 1.9. *Let $\mathcal{M}_\mathcal{A}$ be a boolean valued model, then $[\![\psi \to \phi]\!]$ is $\mathcal{M}_\mathcal{A}$-valid if and only if $[\![\psi]\!] \leq [\![\phi]\!]$.*

Proof. Assume $[\![\psi]\!] \leq [\![\phi]\!]$. Then $[\![\neg\psi \vee \phi]\!] = \neg[\![\psi]\!] \vee [\![\phi]\!] \geq \neg[\![\phi]\!] \vee [\![\phi]\!] = 1_\mathcal{A}$, hence $[\![\neg\psi \vee \phi]\!] = 1_\mathcal{A}$. Now assume $[\![\psi \to \phi]\!]$ is $\mathcal{M}_\mathcal{A}$-valid. Then $[\![\neg\psi \vee \phi]\!] = \neg[\![\psi]\!] \vee [\![\phi]\!] = 1_\mathcal{A}$ which implies $\neg([\![\psi]\!] \wedge \neg[\![\phi]\!]) = 1_\mathcal{A}$, implying $[\![\psi]\!] \wedge \neg[\![\phi]\!] = 0_\mathcal{A}$. But then $[\![\psi]\!] = [\![\psi]\!] \wedge 1_\mathcal{A} = [\![\psi]\!] \wedge ([\![\phi]\!] \vee \neg[\![\phi]\!]) = ([\![\psi]\!] \wedge [\![\phi]\!]) \vee ([\![\psi]\!] \wedge \neg[\![\phi]\!]) = [\![\psi]\!] \wedge [\![\phi]\!]$ so by definition $[\![\psi]\!] \leq [\![\phi]\!]$. □

An easy corollary of this result is that modus ponens preserves validity.

Corollary 1.10. *If* $\llbracket\phi\rrbracket = 1_{\mathcal{A}}$ *and* $\llbracket\phi \to \psi\rrbracket = 1_{\mathcal{A}}$ *hold, then also* $\llbracket\psi\rrbracket = 1_{\mathcal{A}}$

Proof. By the proposition above, $\llbracket\phi \to \psi\rrbracket = 1_{\mathcal{A}}$ implies $\llbracket\phi\rrbracket \leq \llbracket\psi\rrbracket$. But because $\leq$ is a partial order and $\llbracket\phi\rrbracket = 1_{\mathcal{A}} \geq a$ holds for all $a \in A$ this implies $\llbracket\psi\rrbracket = \llbracket\phi\rrbracket = 1_{\mathcal{A}}$. □

Chapter 2

The Forcing Method

We can now present the forcing method described in [33]. Forcing was first introduced by Paul Cohen in his seminal paper “The independence of the continuum hypothesis” [16]. The method presented in this thesis is, however, based on an alternative approach to forcing introduced by Dana Scott [45].

There have been several attempts to use forcing in bounded arithmetic. Notable are the works of Jeff Paris and Alex Wilkie [39], Søren Riis [41], Miklós Ajtai [2], [3], [4] and Albert Atserias and Moritz Müller [6]. However, none of these approaches was as general or pursued as methodically as Krajíček’s forcing method. From now on everything not attributed otherwise can be found in Krajíček’s book[1].

In general, forcing is a method for building a model with some desired properties from a well behaved ‘ground’ model. We start our discussion by fixing a nonstandard model M of true arithmetic as a starting point to construct new models. This model will act more or less as an analog to the ground model in set theoretic forcing. We want M to have some nice model theoretic properties. This is also comparable to the set theoretic case where the ground model is not just any model of ZFC but a countable and transitive model of ZFC.

Notation 2.1. From now on M will be an arbitrary fixed $\aleph_1$-saturated nonstandard model of true arithmetic. We call M our ambient model.

2.1 The boolean algebras

In order to build a boolean valued model we have to fix a boolean algebra. In this section we define not one but two boolean algebras where the first on

[1] At times this text is a lot more detailed than the book. In this case if not attributed otherwise the extra details are due to the author of this text.

acts as a stepping stone to build the second one.

2.1.1 The algebra $\mathcal{A}_\Omega$

The first algebra we want to define is based on a so-called sample space Ω, a definable subset of M. This sample space is not fixed and can be varied depending on the model we want to construct. The process of building the algebra, however, will always be the same.

Definition 2.1. We call a definable subset X of M **nonstandard finite** if there is an $m \in M \setminus \mathbb{N}$ such that $M \models |X| = m$ holds. Now let Ω be a definable nonstandard finite subset of M. We then call Ω a **sample space**.

For the rest of this chapter we use Ω to denote an arbitrary fixed sample space. Using this Ω we define the first boolean algebra $\mathcal{A}_\Omega$.

Definition 2.2. The **structure** $\mathcal{A}_\Omega$ consists of the set of all M-definable subsets of Ω together with interpretations for all boolean operations:

$$\begin{array}{rrr} a \vee b := a \cup b & a \wedge b := a \cap b & \neg a := \overline{a} \\ 1_{\mathcal{A}_\Omega} := \Omega & 0_{\mathcal{A}_\Omega} := \emptyset & \end{array}$$

It is easy to check that this is actually a boolean algebra.

Proposition 2.1. *$\mathcal{A}_\Omega$ is a boolean algebra.*

Proof. The axioms of a boolean algebra hold for $\mathcal{A}_\Omega$ trivially because all the rules are well known general properties of the corresponding set operations.

Furthermore, A_Ω is closed under the boolean operations: Obviously Ω and $\emptyset$ are definable subsets of Ω. The first by definition and the second is defined by $\neg x = x$. Now look at $a, b \in A_\Omega$. There are formulas $\alpha(x)$ and $\beta(x)$ defining a and b by the definition of $\mathcal{A}_\Omega$. But then $a \cup b$ is defined by $\alpha(x) \vee \beta(x)$, $a \cap b$ is defined by $\alpha(x) \wedge \beta(x)$ and $\overline{a}$ is defined by $\neg\alpha(x) \wedge \gamma(x)$ where $\gamma(x)$ defines Ω. □

However, $\mathcal{A}_\Omega$ is in general not a σ-algebra, as can be seen by the following argument.

Example 2.1. We observe first that no infinite subset of $\mathbb{N}$ can be definable in M. Assume otherwise that $N \subseteq \mathbb{N}$ is infinite and definable in M by a formula $\phi(x)$. N is cofinal in $\mathbb{N}$ because it is infinite therefore

$$\psi(k) := \exists x (k < x \wedge \phi(x))$$

holds for all $k \in \mathbb{N}$. However, as $N \subseteq \mathbb{N}$ holds, $\psi(k)$ is not true for any $k \in M \setminus \mathbb{N}$ contradicting overspill (proposition 1.4).

Now fix an arbitrary sample space Ω such that $\Omega \cap \mathbb{N}$ is infinite. Let $\mathcal{A}_\Omega$ be the boolean algebra based on Ω as defined above. As we have seen above $\Omega \cap \mathbb{N}$ can not be definable in M. On the other hand all singletons $\{m\}$ with $m \in \Omega \cap \mathbb{N}$ are definable because there are constants for them in L_{all}. We claim that the set $X := \{\{m\} \mid m \in \Omega \cap \mathbb{N}\} \subseteq A_\Omega$ has no supremum in $\mathcal{A}_\Omega$. Assume otherwise that $a \in A_\Omega$ is a supremum of X. We know $\Omega \cap \mathbb{N} \subsetneq a \subseteq \Omega$ so there is a maximal nonstandard $m^* \in M \setminus \mathbb{N}$ with $m^* \in a$. However,this is a contradiction because then $a \setminus \{m^*\}$ is definable in M and it is easy to see that $\{m\} \leq a \setminus \{m^*\} < a$ holds for all $\{m\} \in X$ contradicting the choice of a.

As we have seen, it is necessary to have a complete boolean algebra to build a boolean valued model, so we are not content with $\mathcal{A}_\Omega$. Fortunately, it is relativity easy to modify $\mathcal{A}_\Omega$ to our satisfaction.

2.1.2 The algebra $\mathcal{B}$ and the measure μ

We want to modify $\mathcal{A}_\Omega$ to get an algebra that ignores 'small' differences between sets. What 'small' means will be determined by an ideal. An ideal is, in set theory, morally speaking, a family of small or negligible sets.

Definition 2.3. Let X be an arbitrary set and $\emptyset \neq \mathcal{I} \subseteq \mathcal{P}(X)$, then $\mathcal{I}$ is called an **ideal** if the following holds

- If $Y \in \mathcal{I}$ and $Z \subseteq Y$ hold, then also $Z \in \mathcal{I}$
- If $Y, Z \in \mathcal{I}$ holds, then also $Y \cup Z \in \mathcal{I}$

We want to use the following measure to determine which elements of $\mathcal{A}_\Omega$ are small, i.e are going to be included in the ideal.

Definition 2.4. In the following we write N for $|\Omega|$ computed in M. Then, the **counting measure** of any $a \in A_\Omega$ is defined to be the nonstandard rational[2] $\frac{|a|}{N}$.

Observe that this measure is not an ordinary measure as it assigns nonstandard rationals instead of reals. This is, however, quite convenient, as the

[2] Of course rationals are not elements of M. However, we can obviously code rationals as pairs (see reminder 1.2). This way, we can treat rationals as elements of $\mathbb{N}$ respectively M. Analogously to integers, rationals in M that have a nonstandard numerator and/or denominator are called nonstandard rationals.

nonstandard rationals do provide a natural choice for negligible values, the infinitesimals.[3]

Definition 2.5. We call a nonstandard rational q an **infinitesimal** if $q < \frac{1}{k}$ holds for all $k \in \mathbb{N}$. We call a standard real x a **standard part** of a nonstandard rational r if $r - x$ is infinitesimal.

This notion allows us to define a family of small sets in $\mathcal{A}_\Omega$ that we can use as an ideal.

Definition 2.6. The **family $\mathcal{I} \subseteq A_\Omega$ of sets with infinitesimal counting measure** is defined by

$$\mathcal{I} := \{a \in A_\Omega \mid \frac{|a|}{N} \text{ is infinitesimal}\}$$

In order to work with this family, we have to prove a few key facts about infinitesimals.

Lemma 2.2.

1. *Let $m \in M \setminus \mathbb{N}$ be a nonstandard number. Then $\frac{1}{m}$ is infinitesimal.*

2. *If r and s are infinitesimals so is $r + s$.*

3. *The standard part x of a nonstandard rational r is unique if it exists.*

4. *If r and s are infinitesimally close (i.e $r - s$ is infinitesimal) they have the same standard part.*

5. *If r is a nonstandard rational such that $n_1 < r < n_2$ holds for $n_1, n_2 \in \mathbb{Z}$ then r has a standard part.*

6. *Let r and s be two nonstandard rationals with standard part x and y respectively. Then the standard part of $r + s$ is $x + y$.*

Proof.

1. $k < l \to \frac{1}{k} > \frac{1}{l}$ is true in $\mathbb{N}$ and therefore also in M. Hence, $m > k$ for all $k \in \mathbb{N}$ implies $\frac{1}{m} < \frac{1}{k}$ for all $k \in \mathbb{N}$.

[3] For the extremely interesting history of the concept of an infinitesimal one can consult, for example, [46].

2. Assume otherwise that $r+s$ is not infinitesimal. All facts about fractions that are true in $\mathbb{N}$, are also true in M. Therefore, we can write $(r+s)-s=r$. But as $r+s$ is not infinitesimal there is a $k_1 \in \mathbb{N}$ such that $r+s>\frac{1}{k_1}$ holds. As s is infinitesimal, we can find a $k_1 > k_2 \in \mathbb{N}$ such that $s<\frac{1}{k_2}$ holds. Therefore, $r>\frac{1}{k_1}-\frac{1}{k_2} \geq \frac{1}{k_1 k_2} \in \mathbb{Q}$ holds which is a contradiction because r is assumed to be infinitesimal.

3. Assume otherwise there are $x \neq y \in \mathbb{R}$ such that $x-r$ and $y-r$ are infinitesimal. Then $(x-r)-(y-r)=x-y$ is also infinitesimal. However, this can not be because $x-y \in \mathbb{R}$ holds which implies immediately that there is a $k \in \mathbb{N}$ such that $x-y>\frac{1}{k}$ holds.

4. Let x be a real and r and s nonstandard rationals. If $x-r$ and $r-s$ are infinitesimal we know by (2) that $x-s=(x-r)+(r-s)$ is infinitesimal, too. Therefore, x is the standard part of s.

5. Assume r has no standard part. Consider the set $X=\{x \in \mathbb{R} \mid r \leq x\}$. This set is not empty because $n_2 \in X$ holds. Furthermore, X is bounded from below as all its elements are bigger than n_1. By the completeness of $\mathbb{R}$ we know that $x_0 := \inf(X) \in \mathbb{R}$ exists. We claim that x_0 is the standard part of r. Otherwise there would be a $k \in \mathbb{N}$ such that $x_0 - r > \frac{1}{k}$ holds. But then $x_0 - \frac{1}{2k} \in \mathbb{R}$ would also be in X contradicting the choice of x_0

6. By (2) $(x+y)-(r+s)=(x-r)+(y-s)$ is infinitesimal.

□

It is important to observe that being infinitesimal is not definable in M. This is an easy application of the overspill concept.

Example 2.2. Assume there is formula $\alpha(x)$ that defines infinitesimal in M. Then $\beta(y) := \forall x(\alpha(x) \rightarrow x < \frac{1}{y})$ defines $\mathbb{N}$. By definition of an infinitesimal $M \models \beta(n)$ holds for all $n \in \mathbb{N}$. On the other hand, for every $m \in M \setminus \mathbb{N}$ its predecessor $m-1$ is nonstandard, hence $\frac{1}{m-1}$ is infinitesimal (lemma 2.2 (1)). Furthermore $\frac{1}{m-1} > \frac{1}{m}$ holds in M, hence $\beta(m)$ does not hold. Contradiction!

The family $\mathcal{I}$ is actually an ideal as we intended.

Proposition 2.3. *The family $\mathcal{I}$ of sets with infinitesimal counting measure is an ideal in A_Ω.*

Proof. First of all, $\mathcal{I}$ is not empty as all singletons have counting measure $\frac{1}{N}$ which is infinitesimal by lemma 2.2(1) because N is nonstandard. Now assume $a \in \mathcal{I}$ and $b \subseteq a$. Then $|b| < |a|$ holds by definition, hence, $\frac{|b|}{N}$ is infinitesimal as $\frac{|b|}{N} < \frac{|a|}{N}$ holds. Finally, assume $a, b \in \mathcal{I}$, then the counting measure of $a \cup b$ is smaller than $\frac{|a|+|b|}{N} = \frac{|a|}{N} + \frac{|b|}{N}$ which is infinitesimal by lemma 2.2 (2). □

Using this ideal we can define an equivalence relation on $\mathcal{A}_\Omega$.

Definition 2.7. We say $a, b \in A$ are **equivalent with respect to** $\mathcal{I}$, if the symmetric difference between a and b is infinitesimal, or in symbols $a \triangle b := (a \cup b) \setminus (a \cap b) \in \mathcal{I}$. In this case we write $a \sim_\mathcal{I} b$.

Observe that this definition makes sense because "\" can be defined using union and complement, so the symmetric difference of a and b is in A_Ω. The fact that this actually is a equivalence relation is also easy to see. Reflexiveness and symmetry are trivial and for transitivity one only needs to observe that $a \triangle c \subseteq a \triangle b \cup b \triangle c$ holds and use the properties of the ideal. Using this equivalence relation, we can build our second boolean algebra.

Definition 2.8. The **structure $\mathcal{B}$ based on Ω** is defined as $\mathcal{B} := \mathcal{A}_\Omega / \mathcal{I}$. In other words $\mathcal{B}$ consist of the set B of all equivalence classes of A_Ω under $\sim_\mathcal{I}$ together with the boolean operations defined by applying the according operations of $\mathcal{A}_\Omega$ on representatives of the equivalence classes in B.

Furthermore, we define the **measure** μ on $\mathcal{B}$ of $b \in B$ to be the standard part of the counting measure of any representative of the class b.[4]

We will dedicate the rest of this section to proving the key facts about $\mathcal{B}$ and μ. The first thing we check is that the definition of μ yields a measure in the measure theoretic sense.

Lemma 2.4. *The measure μ as defined above is a well-defined additive function from B to $[0, 1]$. Furthermore, the measure μ is strict, i.e $\mu(b) = 0$ holds for $b \in B$ if and only if b is the equivalence class of the empty set.*

Proof. μ takes standard values in $[0, 1]$: The counting measure takes nonstandard values only between 0 and 1 because $|a|$ is smaller than N for all $a \in A_\Omega$ and '$k_1 < k_2 \to \frac{k_1}{k_2} < 1$' holds in M as well as '$k_1, k_2 > 0 \to \frac{k_1}{k_2} > 0$'.

Furthermore, the measure μ is well defined: $\mu(b)$ is defined for every $b \in B$ because every nonstandard rational between 0 and 1 has a standard part (lemma 2.2 (5)). Moreover, if two nonstandard rationals are infinitesimally

[4] This measure is also called the Loeb's measure.

close, their standard part is the same (lemma 2.2 (4)). Now obviously if $a_1, a_2 \in A$ are in the same equivalence class, their measures are infinitesimal close. Therefore μ does not depend on the choice of the representative. Finally, the standard part of a nonstandard rational r is unique (lemma 2.2 (3)), hence μ is well defined.

μ is strict: If $\mu(b) = 0$ holds for any $b \in B$, we know that b is the equivalence class of the empty set because the symmetric difference of a set $a \in A$ with the empty set is just a. Hence, if the counting measure of a is infinitesimal, so is the difference to the empty set.

Finally μ is additive: For two nonstandard rationals r, s with standard part x and y the standard part of $r + s$ is $x + y$ (lemma 2.2 (6)). Therefore, the counting measure of $a_1 \cup a_2$ for $a_1, a_2 \in A$ with $a_1 \cap a_2 = \emptyset$ is the sum of the counting measures of a_1 and a_2, since for $b_1, b_2 \in B$ satisfying $b_1 \wedge b_2 = 0_{\mathcal{B}}$ we can pick $a_1 \in b_1$ and $a_2 \in b_2$ such that $a_1 \cap a_2 = \emptyset$ holds. This implies that μ is additive. □

The following theorem can be viewed as the main result of this section. We will use (rather well-known) arguments from both nonstandard analysis and measure theory to prove it

Theorem 2.5. *$\mathcal{B}$ is a complete boolean algebra with the ccc-property and μ is σ-additive.*

Proof. We split the proof in a number of claims:

Claim. *$\mathcal{B}$ is a well-defined boolean algebra.*

The boolean operations are well defined i.e they don't depend on the choice of the representative: Assume $a_1, a_2, a_3, a_4 \in A$, $a_1 \sim a_2$ and $a_3 \sim a_4$ hold. Then

$$(a_1 \cup a_3)\triangle(a_2 \cup a_4) \subseteq (a_1 \triangle a_2) \cup (a_3 \triangle a_4)$$

implies $a_1 \cup a_3 \sim a_2 \cup a_4$ and

$$(a_1 \cap a_3)\triangle(a_2 \cap a_4) \subseteq (a_1 \triangle a_2) \cup (a_3 \triangle a_4)$$

implies $a_1 \cap a_3 \sim a_2 \cap a_4$. We can skip the complement, because it can be defined using $\cup$, $\cap$ and Ω.

Now $\mathbb{B}$ is trivially a boolean algebra because the boolean operations commute with taking the quotient by definition and $\mathbb{A}_\Omega$ is a boolean algebra. For example if $a \in A_\Omega$ represents $b \in B$ we get

$$b \vee 0_{\mathbb{B}} = a/\mathcal{I} \vee \emptyset/\mathcal{I} = (a \cup \emptyset)/\mathcal{I} = a/\mathcal{I} = b$$

The other axioms and the closure properties are proven analogously.

Claim. *$\mathcal{B}$ is a σ-algebra and μ is σ-additive.*

Let $(b_k)_{k\in\mathbb{N}}$ be a countable sequence of elements of B. We fix a sequence $(a_k)_{k\in\mathbb{N}}$ of elements of A_Ω such that a_k is a representative for b_k for all $k \in \mathbb{N}$. First we try to find a supremum for the sequence $(b_k)_{k\in\mathbb{N}}$.

We can assume without loss of generality that $a_0 \subseteq a_1 \subseteq \ldots$ holds because we can pass over to the sequence $a_0, (a_0 \cup a_1), (a_0 \cup a_1 \cup a_2), \ldots$ without changing the set of upper bounds. Then the sequence of the counting measures of the a_k is increasing but bounded (the counting measure is always bounded by 1). This means it is a Cauchy sequence[5]. In other words, for all $k \in \mathbb{N}$ there is a n^* such that for all $n^* < l < m$ we get

$$\frac{|a_l|}{N} \leq \frac{|a_m|}{N} \leq \frac{|a_l|}{N} + \frac{1}{k}$$

We consider the subsequence $(a^*_j)_{j\in\mathbb{N}} := (a_{n_k})_{k\in\mathbb{N}}$ that satisfies for all $k \in \mathbb{N}$ and for all $m > l > k$

$$\frac{|a^*_l|}{N} \leq \frac{|a^*_m|}{N} \leq \frac{|a^*_l|}{N} + \frac{1}{k}$$

Now we use proposition 1.5 to extend $(a^*_j)_{j\in\mathbb{N}}$ to sequence $(a^*_i)_{i<t}$ of nonstandard length. Then proposition 1.4 tells us that for every property of all elements with standard index there is also a element with nonstandard index with this property. Now for every standard s we have

$$`a^*_s \in A_\Omega \wedge \forall i \leq s \ a^*_i \subseteq a^*_s \wedge \frac{|a^*_i|}{N} \leq \frac{|a^*_s|}{N} \leq \frac{|a^*_i|}{N} + \frac{1}{i}'$$

Therefore, we can find a nonstandard s_0 such that $a^*_{s_0}$ has the same properties. This implies for all $k \in \mathbb{N}$ that $a_k \subseteq a^*_{s_0}$ holds. Hence, $a^*_{s_0}$ is an upper bound for $(a_k)_{k\in\mathbb{N}}$. Consequently b defined as the equivalence class of $a^*_{s_0}$ is an upper bound for $(b_k)_{k\in\mathbb{N}}$, because $a_k \leq a^*_{s_0}$ implies $b_k \leq b$ by the definition of $\wedge$ in $\mathcal{B}$.

Now assume b is not the least upper bound of $(b_k)_{k\in\mathbb{N}}$. Then there is a $b' \neq b$ in B such that $b' < b$ and $b_k \leq b'$ holds for all $k \in \mathbb{N}$. Let $a' \in A$ be a representative of b'. $b_k \leq b'$ implies that $a_k \cap a'$ is equivalent to a_k, hence, $(a_k \cap a')\triangle a_k = a_k \setminus a'$ is infinitesimal. Now $\frac{|a^*_{s_0}|}{N} \leq \frac{|a^*_k|}{N} + \frac{1}{i}$ and $a^*_k \subseteq a^*_{s_0}$ for all $k \in \mathbb{N}$ implies $\frac{|a^*_{s_0} \setminus a^*_k|}{N} < \frac{1}{k}$ for all $k \in \mathbb{N}$ because of the additivity of the counting measure. Therefore, again because of the additivity of the counting measure, $a^*_{s_0} \setminus a' \subseteq (a^*_{s_0} \setminus a_k) \cup (a_k \setminus a')$ has counting measure smaller than $\frac{1}{k}$ for all $k \in \mathbb{N}$. Hence, $a^*_{s_0} \setminus a'$ has infinitesimal counting measure. Furthermore

[5] The claim ‘bounded increasing sequences are Cauchy sequences’ is true and first order in $\mathbb{N}$ and hence true in M

$a' \setminus a^*_{s_0}$ has infinitesimal counting measure because $b' \leq b$ holds. Therefore, $a^*_{s_0} \triangle a'$ has infinitesimal counting measure. But this would imply $b = b'$, a contradiction to $b' < b$. Hence, $\mathcal{B}$ is a σ-algebra.

Furthermore, $\frac{|a^*_i|}{N} \leq \frac{|a^*_{s_0}|}{N} \leq \frac{|a^*_i|}{N} + \frac{1}{i}$ implies that the limsup and the liminf of the counting measures of $(a_j)_{j\in\mathbb{N}}$ are bounded infinitesimally close to the counting measure of $a^*_{s_0}$, hence the standard parts of these counting measures, i.e $\mu(b_j)$ convergences to the standard part of the counting measure of $a^*_{s_0}$, i.e $\mu(b)$. Now assume the b_k are pairwise disjoint for all $k \in \mathbb{N}$. Then the sequence $b^*_k = \bigvee_{i\leq k} b_i$ has the property that $\mu(b^*_k) = \Sigma_{i\leq k}\mu(b_i)$ holds by the (finite) additivity of μ. Furthermore any upper bound for $(b_k)_{k\in\mathbb{N}}$ is also a upper bound of $(b^*_k)_{k\in\mathbb{N}}$, hence $b = \bigvee_{i\in\mathbb{N}}(b^*_i)$ holds. Therefore the following holds

$$\mu(b) = \lim_{i\to\infty}\mu(b^*_i) = \lim_{i\to\infty}(\Sigma_{j\leq i}\mu(b_j)) = \Sigma^{\infty}_{j=1}\mu(b_j)$$

Hence, μ is σ-additive.

Claim. *$\mathcal{B}$ has the ccc-property.*

This is a rather easy consequence of the strictness of the measure (see lemma 2.4) together with the σ-additivity[6]. Look at the following partitioning $P := \{(\frac{1}{n+1}, \frac{1}{n}] \mid n \in \mathbb{N} \setminus \{0\}\}$ of $(0, 1] \subseteq \mathbb{R}$. For every interval $(\frac{1}{n+1}, \frac{1}{n}]$ in P let B_n be the set of all elements b of B with measure $\mu(b) \in (\frac{1}{n+1}, \frac{1}{n}]$. Then every B_n contains only finitely many disjunct elements (i.e. $a \wedge b = 0_{\mathcal{B}}$ holds only for finitely many elements). Otherwise by additivity of μ we would get a set with measure larger 1, but this contradicts lemma 2.4. As there are only countably many B_n and $\bigcup_{k\in\mathbb{N}} B_k = B \setminus \{0_{\mathcal{B}}\}$ holds, we know that every antichain contains only countably many nonzero elements.

Claim. *Every family F of elements of B has a countable subfamily with the same set of upper bounds as F.*

Let $\mathcal{I}(F)$ be the (downwards) closure of F under $\leq$ and $\vee$. Then $\mathcal{I}(F)$ is the so-called ideal generated by F. Take a maximal antichain X in $\mathcal{I}(F)$. We claim that X and F share the same set of upper bounds.

First we show that every upper bound for X is also an upper bound for $\mathcal{I}(F)$ hence also an upper bound for its subset F. Assume otherwise that there is a $b \in B$ with $b \geq x$ for all $x \in X$ but $b \ngeq i$ for some $i \in \mathcal{I}(F)$. This implies $i \wedge \neg b \neq 0_{\mathcal{B}}$ (otherwise $i = i \wedge 1_{\mathcal{B}} = i \wedge (b \vee \neg b) = (i \wedge b) \vee (i \wedge \neg b) = i \wedge b$ would contradict $i \nleq b$). Furthermore, $x \leq b$ implies $x \wedge \neg b = 0_{\mathcal{B}}$ so $x \wedge \neg b \wedge i = 0_{\mathcal{B}}$ holds for all $x \in X$ which implies $i \wedge \neg b \notin X$. Since $\mathcal{I}(F)$ is

[6] The connection between the ccc-property and strict measures is long known and has been thoroughly investigated see e.g [24]

downwards closed under $\leq$ we know that $i \wedge \neg b \leq i$ implies $i \wedge \neg b \in \mathcal{I}(F)$. But then X is not a maximal antichain in $\mathcal{I}(F)$ contradicting its choice.

On the other hand because of $X \subseteq \mathcal{I}(F)$ for every $x \in X$, by definition of $\mathcal{I}(F)$, we can fix finitely many $f_1^x, f_2^x, \dots f_k^x \in F$ such that $\bigvee_{i \leq k} f_i^x \geq x$ holds. Therefore, any upper bound for F is also an upper bound for X. This means X and F have the same upper bounds.

Now let F_0 be the family defined by $F_0 := \{f_k^x \mid x \in X\}$ with f_k^x as above. Then, every upper bound of F is an upper bound of F_0 because $F_0 \subseteq F$ holds and every upper bound of F_0 is an upper bound of X by definition so its again an upper bound of F as we have shown. This means F and F_0 have the same upper bounds. Furthermore, X is countable because it is a antichain and $\mathcal{B}$ has the ccc-property. But this implies that F_0 is also countable, hence we found a set with the desired properties.

Claim. *$\mathcal{B}$ is complete.*

Let F be a family of elements of B. Then there is a countable F_0 with the same set of upper bounds. F_0 has a supremum $b \in B$ because $\mathcal{B}$ is a σ-algebra. But then b is trivially also a supremum of F. □

This concludes our introduction of the boolean algebras $\mathcal{A}_\Omega$ and $\mathcal{B}$.

2.2 Building the Model $K(F)$

In this section we intend to build a boolean valued model $K(F)$ based on the boolean algebra $\mathcal{B}$. Subsequently we will discuss easy properties of $K(F)$ and finally we will look at the connections between $K(F)$ and the measure μ on $\mathcal{B}$.

2.2.1 Defining $K(F)$

We can now start to force with random variables, i.e to produce a boolean valued model $K(F)$ from M, Ω and $\mathcal{B}$. The universe of $K(F)$ will be a set of well behaved functions which we call random variables.

Definition 2.9. Let Ω be a sample space. A nonempty set F of functions is called an **family of random variables** if the following holds

- $\alpha : \Omega \rightarrow M$ for all $\alpha \in F$.
- All $\alpha \in F$ are M-definable.

In general, we don't want to construct an L_{all}-model. Instead we will pick a different fragment L of L_{all} for every application. In order to build an L-model we need the family of random variables to be 'closed' in relation to this L.

Definition 2.10. Let $L \subseteq L_{all}$ be a language, Ω a sample space and F a family of random variables. We say F is **L-closed** if F contains all L-constants and is closed under L-functions, i.e for every $f \in L$ and $\alpha_1, \alpha_2, \ldots \alpha_k \in F$ there is a $\alpha \in F$ such that

$$\forall \omega \in \Omega \; f(\alpha_1(\omega), \alpha_2(\omega), \ldots \alpha_k(\omega)) = \alpha(\omega)$$

holds in M.

Given the boolean algebra $\mathcal{B}$ over some sample space Ω, a language L and an L-closed family of random variables F we can now define the model $K(F)$. Observe that $\mathcal{B}$ is completely determined by Ω which can be read off F. Therefore, writing $K(F)$ is unambiguous for a fixed language L.

Definition 2.11. Let $L \subseteq L_{all}$ be a language and F an L-closed family of random variables over a sample space Ω. Then the **model** $K(F)$ is defined to be the boolean valued L-model over the boolean algebra $\mathcal{B}$ based on Ω with universe F that interprets every L-function f by

$$f(\alpha_1, \alpha_2 \ldots \alpha_k)(\omega) := f(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))$$

and valuates atomic formulas by the following rules

- $[\![\alpha = \beta]\!] := \{\omega \in \Omega \mid \alpha(\omega) = \beta(\omega)\}/I$
- $[\![R(\alpha_1, \alpha_2 \ldots \alpha_k)(\omega)]\!] := \{\omega \in \Omega \mid R(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))\}/I$

2.2.2 Easy properties of $K(F)$

First we prove a useful lemma that gives us an easy way to represent the truth value of an quantifier free sentence via an element of $\mathcal{A}_\Omega$.

Lemma 2.6. *Let $\phi(x_1, \ldots x_k)$ be a quantifier free L-formula and $\alpha_1, \ldots \alpha_k \in F$ random variables. Then $\{\omega \in \Omega \mid \phi(\alpha_1(\omega), \ldots \alpha_k(\omega)\}$ is a representative of the equivalence class $[\![\phi(\alpha_1, \ldots \alpha_k)]\!]$.*

Proof. In the following we write $\overline{\alpha}(\omega)$ for $(\alpha_1(\omega), \ldots \alpha_k(\omega))$. The proof is by induction on the formula complexity. For atomic formulas $\{\omega \in \Omega \mid \phi(\overline{\alpha}(\omega))\}$ is a representative of $[\![\phi(\overline{\alpha})]\!]$ by definition. So assume $\phi = \xi \wedge \psi$.

Then $[\![\phi(\overline{\alpha})]\!] = [\![\xi(\overline{\alpha})]\!] \wedge [\![\psi(\overline{\alpha}]\!]$, hence by definition of $\mathcal{B}$ and the inductive assumption we know that $[\![\phi(\overline{\alpha})]\!]$ is represented by

$$\{\omega \in \Omega \mid \xi(\overline{\alpha}(\omega))\} \cap \{\omega \in \Omega \mid \psi(\overline{\alpha}(\omega))\} = \{\omega \in \Omega \mid \xi(\overline{\alpha}(\omega)) \wedge \psi(\overline{\alpha}(\omega))\}$$

The cases $\phi = \xi \vee \psi$ and $\phi = \neg\xi$ are treated analogously. □

The next propositions show that we have defined a reasonable model.

Proposition 2.7. *All axioms of first order logic are $K(F)$-valid.*

Proof (Sketch). There is no definitive list of axioms for first order logic but many possible sets of axioms. However, the statement above is true for every possible axiomatization. We don't fix a complete list of axioms, instead we will just show three examples that highlight the three lines of argumentation that can be used for most commonly used axioms, hopefully enabling the reader to check his favorite list of axioms himself.

- *Law of the excluded middle:* $[\![\phi \vee \neg\phi]\!] = [\![\phi]\!] \vee \neg[\![\phi]\!] = 1_{\mathcal{B}}$ by the axioms of a boolean algebra and the definition of a boolean valued model.

- *Transitivity of Equality:* Assume for $\alpha, \beta, \gamma \in F$ that $[\![\alpha = \beta]\!] = 1_{\mathcal{B}}$ and $[\![\beta = \gamma]\!] = 1_{\mathcal{B}}$ hold. Then $\{\omega \in \Omega \mid \alpha(\omega) \neq \beta(\omega)\}$ and $\{\omega \in \Omega \mid \beta(\omega) \neq \gamma(\omega)\}$ have infinitesimally small counting measure. But then $\{\omega \in \Omega \mid \alpha(\omega) \neq \gamma(\omega)\} \subseteq \{\omega \in \Omega \mid \alpha(\omega) \neq \beta(\omega)\} \cup \{\omega \in \Omega \mid \beta(\omega) \neq \gamma(\omega)\}$ has infinitesimal counting measure. Now this implies $\{\omega \in \Omega \mid \alpha(\omega) = \gamma(\omega)\} \sim_{\mathcal{I}} \Omega$ according to definition 2.7 or in other words $[\![\alpha = \gamma]\!] = 1_{\mathcal{B}}$

- *Quantifier and Negation* We want to show $[\![\neg\forall x\phi(x)]\!] = [\![\exists x\neg\phi(x)]\!]$. We know
$$[\![\neg\forall x\phi(x)]\!] = \neg \inf_{\alpha\in F}[\![\phi(\alpha)]\!]. =: \neg b$$
for some $b \in B$. Then $\neg b \leq [\![\phi(\alpha)]\!]$ holds by definition for all $\alpha \in F$. Furthermore, $b' \leq [\![\phi(\alpha)]\!]$ for all $\alpha \in F$ implies $b' \leq \neg b$ for all $b' \in B$. Therefore, we know
$$b \geq \neg[\![\phi(\alpha)]\!] = [\![\neg\phi(\alpha)]\!]$$
for all $\alpha \in F$ and for all $b' \in B$ with $\neg b' \geq \neg[\![\phi(\alpha)]\!]$ for all $\alpha \in F$ we get $\neg b' \geq b$. This implies
$$b = \sup_{\alpha\in F}[\![\neg\phi(\alpha)]\!] = [\![\exists x\neg\phi(x)]\!]$$
because for every b^* there is a b' such that $\neg b' = b^*$ holds (namely $\neg b^*$). Thus we are done.

□

This entails together, with proposition 2.7, that every first order tautology is valid in $K(F)$. More importantly we can show that $K(F)$ can be used to prove independence results.

Proposition 2.8. *Let T be a first order theory and ϕ a first order formula. Furthermore let $\vdash$ be the entailment relation for an arbitrary complete Hilbert style proof system*[7]*. If T is $K(F)$-valid (i.e every element of T is $K(F)$-valid) and $T \vdash \phi$ holds then ϕ is $K(F)$-valid.*

Proof. A Hilbert style proof is a sequence of formulas ending with ϕ where every element is either a tautology, an element of T or a derived from the earlier elements via modus ponens. Now, as T is $K(F)$-valid and all tautologies are $K(F)$-valid and modus ponens preserves $K(F)$-validity (corollary 1.10), every element of the sequence is $K(F)$-valid, which means also that ϕ is $K(F)$-valid. □

A useful corollary that will be used frequently without mentioning is the following.

Corollary 2.9. *If $\vdash \phi \leftrightarrow \psi$ holds, then $[\![\phi]\!] = [\![\psi]\!]$*

Proof. $\vdash \phi \leftrightarrow \psi$ implies by proposition 2.8 that $[\![\phi \leftrightarrow \psi]\!] = 1_{\mathcal{B}}$ holds, hence proposition 1.9 entails $[\![\phi]\!] \leq [\![\psi]\!]$ and vice versa, hence $[\![\phi]\!] = [\![\psi]\!]$ holds. □

In set theory, if we start with a model of ZFC, the forcing extension will be a model of ZFC, too. Unfortunately no comparably strong statement is true for the method presented here. However, $K(F)$ models at least some weak arithmetic.

Lemma 2.10. *Let $L \subseteq L_{all}$ be a language and F an L-closed family of random variables. Then, all universal L-sentences true in $\mathbb{N}$ are $K(F)$-valid. Furthermore, if F contains constants functions for all $n \in \mathbb{N}$, every $\exists\forall$-sentence true in $\mathbb{N}$ is $K(F)$ valid.*

Proof. Assume ϕ $\mathbb{N} \models \forall x_1 \dots \forall x_n \phi(x_1 \dots x_n)$ holds for a quantifier free formula which also implies $M \models \forall x_1 \dots \forall x_n \phi(x_1, \dots x_n)$. We assume for the sake of contradiction that $K(F) \models \forall x_1 \dots \forall x_n \phi(x_1, \dots x_n)$ doesn't hold. This assumption would imply that there are random variables $\alpha_1, \dots \alpha_n \in F$ such that $[\![\phi(\alpha_1, \dots \alpha_n)]\!] \neq 1_{\mathcal{B}}$ holds. Using lemma 2.6 we get from this that $\{\omega \in \Omega \mid \phi(\alpha_1(\omega), \dots \alpha_n(\omega))\}$ is not equivalent to Ω, hence there is a $\omega^* \in \Omega$

[7] It is irrelevant which proof system is chosen, as long as it is complete.

such that $\phi(\alpha_1(\omega^*), \dots \alpha_n(\omega^*))$ is not true in M contradicting the assumption.

Now assume F contains constants $\overline{1}, \overline{2}, \dots \overline{n} \dots$ for all natural numbers and $\mathbb{N} \models \exists x_1 \dots \exists x_l \forall y_1 \dots \forall y_m \phi(x_1 \dots x_l, y_1, \dots y_m)$ holds. Then there is a tuple of natural numbers $(n_1, \dots n_l) \in \mathbb{N}^l$ such that

$$\mathbb{N} \models \forall y_1 \dots \forall y_m \phi(n_1 \dots n_l, y_1, \dots y_m)$$

holds. Now this is a universal formula and one easily sees that the argument above can easily be adapted to prove

$$K(F) \models \forall y_1 \dots \forall y_m \phi(\overline{n_1} \dots \overline{n_l}, y_1, \dots y_m)$$

. □

On the one hand this result is very useful to show that $K(F)$ models some weak theory of arithmetic. On the other hand it seems to prevent us from proving independence for universal sentences. However, the language L is not fixed, and which statements are universal depends on the language.

2.2.3 μ, $K(F)$ and Probabilities

The next topic we want to discuss is, whether the measure μ on $\mathcal{B}$ has some useful connections to the truth values in $K(F)$. It turns out that at least the propositional logical connectives are compatible with the measure.

Lemma 2.11. *Let ϕ and ψ be L-sentences and μ the measure on $\mathcal{B}$ defined in definition 2.7. Then the following holds:*

- $0 \leq \mu(\llbracket \phi \rrbracket) \leq 1$
- $\mu(\llbracket \neg\phi \rrbracket) = 1 - \mu(\llbracket \phi \rrbracket)$
- $\mu(\llbracket \phi \wedge \psi \rrbracket) + \mu(\llbracket \phi \vee \psi \rrbracket) = \mu(\llbracket \phi \rrbracket) + \mu(\llbracket \psi \rrbracket)$

Proof. The first statement is clear because μ takes only values between 0 and 1. The second statement follows from the additivity of μ and the definition of $\llbracket \dots \rrbracket$ if we consider the following equation

$$1 = \mu(1_{\mathcal{B}}) = \mu(\llbracket \phi \rrbracket \vee \neg \llbracket \phi \rrbracket) = \mu(\llbracket \phi \rrbracket) + \mu(\llbracket \neg\phi \rrbracket)$$

Finally for the third statement consider the following computation using additivity, basic manipulation of formulas and statement two:

$$\begin{aligned}\mu([\![\phi]\!]) + \mu([\![\psi]\!]) = \mu([\![\phi \wedge \neg\psi]\!] \vee [\![\phi \wedge \psi]\!]) + \mu([\![\neg\phi \wedge \psi]\!] \vee [\![\phi \wedge \psi]\!]) = \\ \mu([\![\phi \wedge \neg\psi]\!]) + \mu([\![\neg\phi \wedge \psi]\!]) + 2\mu([\![\phi \wedge \psi]\!]) = \\ \mu([\![(\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi)]\!]) + 2\mu([\![\phi \wedge \psi]\!]) = \\ \mu([\![(\phi \vee \psi) \wedge \neg(\phi \wedge \psi)]\!]) + 2\mu([\![\phi \wedge \psi]\!]) = \\ \mu([\![\neg(\neg(\phi \vee \psi) \vee (\phi \wedge \psi))]\!]) + 2\mu([\![\phi \wedge \psi]\!]) = \\ 1 - (\mu([\![\neg(\phi \vee \psi)]\!]) + \mu([\![\phi \wedge \psi]\!])) + 2\mu([\![\phi \wedge \psi]\!]) = \\ 1 - (1 - \mu([\![\phi \vee \psi]\!])) + \mu([\![\phi \wedge \psi]\!]) = \mu([\![\phi \vee \psi]\!]) + \mu([\![\phi \wedge \psi]\!])\end{aligned}$$

□

There is a deep connection between the probability of truth for a sentence and μ. This is the next topic we want to explore.

Definition 2.12. Let $\phi(x_1, x_2, \ldots x_k)$ be a L-formula and $\alpha_1, \alpha_2, \ldots \alpha_k$ elements of F. Then we denote by

$$\mathbf{Prob}_{\omega\in\Omega}[\phi(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))]$$

the probability that $\phi(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))$ is true for a random $\omega \in \Omega$ under the discrete uniform distribution[8] over Ω. In other words

$$Prob_{\omega\in\Omega}[\phi(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))] := \frac{|\{\omega \in \Omega | \phi(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))\}|}{|\Omega|}$$

For atomic sentences this is basically ‘the counting measure of the sentence’ so the following lemma shouldn’t be too surprising.

Lemma 2.12. *For all $\epsilon > 0$ standard and all quantifier free sentences ϕ the following holds:*

$$Prob_{\omega\in\Omega}[\phi(\alpha_1(\omega), \alpha_2(\omega) \ldots \alpha_k(\omega))] \geq \mu([\![\phi(\alpha_1, \alpha_2 \ldots \alpha_k)]\!]) - \epsilon$$

Proof. By lemma 2.6 we know that $\{\omega \in \Omega \mid \phi(\overline{\alpha}(\omega))\}$ is a representative of $[\![\phi(\overline{\alpha})]\!]$. This implies that $\mu([\![\phi(\overline{\alpha})]\!])$ is the standard part of the counting measure of the set above, which coincides with the probability of $\phi(\overline{\alpha}(\omega))$, hence the inequality follows from the definition of a standard part. □

[8] Observe that this probability is defined in M and $|\Omega|$ is finite in M so choosing this distribution makes sense

Can we improve this lemma to include sentences with quantifiers? The first thing to notice is that we can't extend the proof above to work for quantified sentences because it makes heavy use of the fact that taking quotients commutes with the propositional connectives. However, this is not true for the supremum or infimum. For example,

$$(\bigcup_i a_i)/\mathcal{I} \neq \bigvee_i (a_i/\mathcal{I})$$

may happen for $a_i \in A_\Omega$ even if the left side is defined, e.g if all the a_i are singletons. And actually we can't prove a similar statement for quantified sentences in general as the following example shows.

Example 2.3. Let $\phi(x, y)$ be an atomic L-formula such that $\forall x \exists! y \neg\phi(x, y)$ holds in M. In other words for every possible value $a \in M$ for x there is exactly one value $b \in M$ for y such that $\phi(a, b)$ is not true in M. We call b the counterexample for a. Now assume that there is no function $\alpha \in F$ such that $\alpha(\omega)$ is the counterexample to ω for more than an infinitesimal fraction of possible $\omega \in \Omega$.

Under these assumptions we get $Prob_{\omega\in\Omega}[\forall y\phi(\omega, y)] = 0$ because for all $\omega \in \Omega$ the sentence $\forall y\phi(\omega, y)$ is wrong in M as there is a counterexample b for every $\omega \in \Omega$.

On the other hand $\mu([\![\forall y\phi(id_\Omega, y)]\!]) = 1$ holds because for every $\alpha \in F$ we know $[\![\phi(id_\Omega, \alpha)]\!] = 1_\mathcal{B}$ because $\phi(id_\Omega(\omega), \alpha(\omega))$ is true for all $\omega \in \Omega$ where $\alpha(\omega)$ is not the counterexample to ω. By assumption these are all but an infinitesimal fraction. Thus the infimum of the values $[\![\phi(id_\Omega, \alpha)]\!]$ for all possible $\alpha \in F$ is $1_\mathcal{B}$.

This example relies on the fact that F has no function to find the counterexample sufficiently often . This motivates us to look for conditional generalizations of lemma 2.12 using assumptions about the family F.

2.3 Witnessing Quantifiers

In classical logic the concept of witnessing quantifiers is basically trivial. If $M \models \exists x\phi(x)$ holds, we can find an element $m \in M$ such that $M \models \phi(m)$ holds. For boolean valued logic this need not be the case. In general there can be a (possibly uncountable) sequence $(m_i)_{i\in I}$ of elements of M such that the sequence $([\![\phi(m_i)]\!])_{i\in I}$ converges to $1_\mathcal{B}$ but no element $m \in M$ satisfies $[\![\phi(m)]\!] = 1_\mathcal{B}$. Then $[\![\exists x\phi(x)]\!]$ is valid but there may be no witness for this validity. The goal of this section is, to find conditions under which we can guarantee the existents of witnesses for quantified sentences.

The first thing to note is that our model $K(F)$ has, what could be called, a countable witnessing property. For every valid existential statement there is a countable sequence "witnessing" this validity. This is a consequence of the ccc-property of $\mathcal{B}$.

Lemma 2.13. *For every L-sentence of the form $\exists x\phi(x)$, where $\phi(x)$ is an quantifier free formula, there are countably many $\alpha_k \in F$, $k \in \mathbb{N}$ such that the following holds*

$$[\![\exists x\phi(x)]\!] = \bigvee_{k\in\mathbb{N}} [\![\phi(\alpha_k)]\!]$$

Proof. This follows immediately from the one of the claims in the proof of theorem 2.5, namely that every family of elements of B contains a countable subfamily with the same upper bounds. Therefore, the family $\{[\![\phi(\alpha)]\!] \mid \alpha \in F\}$ has a countable subfamily $\{[\![\phi(\alpha_k)]\!] \mid k \in \mathbb{N}\}$ such that

$$[\![\exists x\phi(x)]\!] = \sup_{\alpha\in F}[\![\phi(\alpha)]\!] = \sup_{k\in\mathbb{N}}[\![\phi(\alpha_k)]\!] = \bigvee_{k\in\mathbb{N}} [\![\phi(\alpha_k)]\!]$$

holds. □

2.3.1 Witnessing for definable families closed under definition by cases

First we consider families that are closed under definable case distinctions. This property will help us to build a single witness from the countable sequence above.

Definition 2.13. We call a family of random variables F **closed under definition by cases** if for all $\alpha_1, \alpha_2 \in F$ and every definable $X \subseteq \Omega$ there is a $\beta \in F$ with the following property

$$\beta(\omega) := \begin{cases} \alpha_1(\omega) & \text{if } \omega \in X \\ \alpha_2(\omega) & \text{else} \end{cases}$$

In order to prove the existence of a witness we also have to assume that the family of random variables is definable. Under this assumptions we can prove the following somewhat technical but also very useful lemma, another application of the $\aleph_1$-saturation of M.

Lemma 2.14. *Let F be a M-definable family of random variables, let $(b_k)_{k\in\mathbb{N}}$ be an antichain in $\mathcal{B}$ and for all $k \in \mathbb{N}$ let $a_k \in A$ be a representative of b_k. Now assume there are $\alpha_k \in F$ such that for all $k \in \mathbb{N}$ there is a $\beta_k \in F$ that satisfies for every $l \leq k$ the equation $\beta_k(\omega) = \alpha_l(\omega)$ for all $\omega \in a_l \setminus \bigcup_{i<l} a_i$. Then there is also a $\beta \in F$ such that $[\![\beta = \alpha_k]\!] \geq b_k$ holds for all $k \in \mathbb{N}$.*

Proof. For every $k \in \mathbb{N}$ let C_k be the set of all $\beta \in F$ such that for all $l \leq k$ we know that $\beta(\omega) = \alpha_l(\omega)$ holds for all $\omega \in a_l \setminus \bigcup_{i<l} a_i$. We note that C_k is definable in M from the finitely many (definable) parameters a_l and α_l for $l \leq k$ because F is definable in M. Furthermore, we note that $\bigcap_{i<k} C_i = C_k$ holds by the definition of C_k. Finally $C_k \neq \emptyset$ is true by assumption so we can apply proposition 1.6 to find a element $\beta^* \in \bigcap_{i\in\mathbb{N}} C_i$. This β^* obviously satisfies $[\![\beta^* = \alpha_k]\!] \geq b_k$ by the choice of a_k. □

Using this lemma we can prove our first witnessing theorem. In [33] this theorem is proven for arbitrary sentences. We will only be interested in witnessing for existential sentences therefore we omit this generalization. It can easily be proven by induction on the number of quantifiers.

Theorem 2.15. *Let F be an M-definable family of random variables that is closed under definition by cases. Then, for every sentence of the form $\phi = \exists x\psi(x)$ where $\psi(x)$ is quantifier free there is a $\alpha \in F$ such that $[\![\exists x\psi(x)]\!] = [\![\psi(\alpha)]\!]$ holds.*

Proof. First we use lemma 2.13 to find a countable family $\alpha_1, \alpha_2 \ldots \alpha_k \ldots$ of elements of F such that $[\![\exists x\psi(x)]\!] = \bigvee_{k\in\mathbb{N}}[\![\psi(\alpha_k)]\!]$ holds. From this we can define a sequence of elements of B by $b_k := [\![\psi(\alpha_k)]\!] \setminus \bigvee_{l<k}[\![\psi(\alpha_l)]\!]$ for all $k \in \mathbb{N}$. It is easy to see that $\bigvee_{i\leq k} b_i = \bigvee_{i\leq k}[\![\psi(\alpha_i)]\!]$ holds and consequently by induction also $\bigvee_{i\in\mathbb{N}} b_i = \bigvee_{i\in\mathbb{N}}[\![\psi(\alpha_i)]\!] = \bigvee_{\alpha\in F}[\![\psi(\alpha)]\!]$.

By definition we know that the b_k form an antichain in $\mathcal{B}$. Furthermore, for a sequence $(a_k)_{k\in\mathbb{N}}$ where a_i is a representative of b_i for every $i \in \mathbb{N}$, we can find for every $k \in \mathbb{N}$ a $\beta_k \in F$ with the property that for all $l \leq k$ we have $\beta_k(\omega) = \alpha_l(\omega)$ for all $\omega \in a_l \setminus \bigcup\limits_{i<l} a_i$ because we can write β_k as

$$\beta_k(\omega) := \begin{cases} \alpha_1(\omega) & \text{if } \omega \in a_1 \\ \alpha_2(\omega) & \text{if } \omega \in a_2 \setminus a_1 \\ & \vdots \\ \alpha_k(\omega) & \text{if } \omega \in a_k \setminus \bigcup\limits_{i<k} a_i \end{cases}$$

and F is closed under definition by cases. Therefore, we can apply lemma 2.14 to find a $\beta \in F$ such that $[\![\beta = \alpha_k]\!] \geq b_k$ holds for every $k \in \mathbb{N}$. Furthermore, by the construction of b_k we know that even $b_k \leq [\![\beta = \alpha_k]\!] \wedge [\![\psi(\alpha_k)]\!]$ is true which means also $b_k \leq [\![\beta = \alpha_k \wedge \psi(\alpha_k)]\!] \leq [\![\psi(\beta)]\!]$. Therefore, the following equation holds

$$[\![\psi(\beta)]\!] \leq [\![\exists x\psi(x)]\!] = \bigvee_{\alpha\in F} [\![\psi(\alpha)]\!] = \bigvee_{k\in\mathbb{N}} b_k \leq [\![\psi(\beta)]\!]$$

which proves that β is the witness we were looking for. □

This witnessing theorem has only one downside. In all applications, the families of random variables used are neither definable nor closed under definition by cases. These requirements are too strong. Therefore, we have to weaken the assumptions of the theorem. However, then we won't be able to prove a similarly strong theorem but only a form of approximative witnessing.

2.3.2 Witnessing for families closed under definition by cases by quantifier free L-formulas

Probably the natural way to weaken the requirement of being closed under definition by cases is to weaken the formulas allowed in the case distinction.

Definition 2.14. We say F is **closed under definition by cases by quantifier free L-formulas** if for all α and β in F and every quantifier free L-formula $\phi(x)$ with parameters from F, the function γ defined by

$$\gamma(\omega) := \begin{cases} \alpha(\omega) & \text{if } \phi(\alpha(\omega)) \text{ holds} \\ \beta(\omega) & \text{else} \end{cases}$$

is in F.

We will see later on that the fact that no quantifiers are allowed in ϕ will be important for the applicability of witnessing to polynomial time random variables. Furthermore, the fact that the case distinction is done with respect to $\phi(\alpha(\omega))$ can be helpful because even though α is M-definable, this definition may not be expressible in L or need quantifiers in L.

Furthermore we don't want to assume that the family F is definable.

Theorem 2.16. *Let F be an L-closed family that is closed under definition by cases by quantifier free L-formulas and let $\phi(x)$ be an quantifier free L-formula (possibly with parameters from F). If $\exists x \phi(x)$ is $K(F)$-valid then for every standard $\epsilon > 0$ there is a $\alpha \in F$ such that $\mu([\![\phi(\alpha)]\!]) > 1 - \epsilon$ holds.*

Proof. Using lemma 2.13 we can find a countable sequence $(\alpha_k)_{k\in\mathbb{N}}$ of random variables with the following property:

$$[\![\exists x \phi(x)]\!] = \bigvee_{k\in\mathbb{N}} [\![\phi(\alpha_k)]\!]$$

Using this sequence we define a second sequence $(\alpha_k^*)_{k\in\mathbb{N}}$ inductively by $\alpha_1^* = \alpha_1$ and

$$\alpha_{i+1}^*(\omega) := \begin{cases} \alpha_i^*(\omega) & \text{if } \phi(\alpha_i^*(\omega)) \text{ holds} \\ \alpha_{i+1}(\omega) & \text{else} \end{cases}$$

First of all, α^*_{i+1} is an element of F for all $i \in \mathbb{N}$ because this is a definition by cases by a quantifier free L-formula and α_{i+1} and, by induction, α^*_i are elements of F. Secondly, it is easy to see that this sequence has the property

$$[\![\phi(\alpha^*_{i+1})]\!] = [\![\phi(\alpha^*_i)]\!] \vee [\![\phi(\alpha_{i+1})]\!]$$

Which gives us by induction

$$[\![\phi(\alpha^*_{i+1})]\!] = \bigvee_{l \leq i+1} [\![\phi(\alpha_l)]\!]$$

Therefore, we know

$$[\![\exists x \phi(x)]\!] = \bigvee_{k \in \mathbb{N}} [\![\phi(\alpha^*_k)]\!]$$

And, furthermore, also for all $i \in \mathbb{N}$

$$[\![\phi(\alpha^*_i)]\!] \leq [\![\phi(\alpha^*_{i+1})]\!]$$

Which implies, using the σ-additivity of μ,

$$lim_{k \to \infty} \mu([\![\phi(\alpha^*_k)]\!]) = \mu([\![\exists x \phi(x)]\!])$$

But because $[\![\exists x \phi(x)]\!]$ is valid by assumption and by the very definition of a limes in $\mathbb{R}$ this means for every standard $\epsilon > 0$ there is a $k \in \mathbb{N}$ such that $\mu([\![\alpha^*_k]\!]) > 1 - \epsilon$ holds

□

Again a more general version of this theorem is proven in [33]. The proof of the generalization is however not as trivial as for the first witnessing theorem. Additionally [33] contains a third witnessing theorem for so-called compact families. We have no use for this theorem, therefore we omit it.

2.3.3 The family of polynomial time functions

We can use coding to treat algorithms as objects of $\mathbb{N}$ and M. First of all we can obviously code strings as natural numbers (we assume in the following that a non surjective encoding is fixed). Furthermore, we can code algorithms as finite sequences of commands. As we have seen in reminder 1.2 we can code these finite sequences as natural numbers hence we can code algorithms as natural numbers. On these codes we can perform the following two operations using L_{all}-functions: We can evaluate an algorithm $\mathbb{A}$ on a string x using the function $eval(\mathbb{A}, x)$ that maps $\mathbb{A}$ and x on the output of $\mathbb{A}$ on x if it exists and on a natural number not coding a string otherwise. We write as a shorthand $\mathbb{A}(x) := eval(\mathbb{A}, x)$. Furthermore, the function $t_{\mathbb{A}}(x)$ maps $\mathbb{A}$ and x on the running time of $\mathbb{A}$ on x. Using these two functions we can easily define polynomial time functions in $\mathbb{N}$.

Definition 2.15. A definable function $f : \{0,1\}^* \to \{0,1\}^*$ is **polynomial time** or short **ptime** if there exists an algorithm $\mathbb{A}$ such that the following holds for some $k \in \mathbb{N}$

$$\mathbb{N} \models \forall x \in \{0,1\}^* \ ((\mathbb{A}(x) = f(x)) \wedge (\ t_{\mathbb{A}}(x) \leq |x|^k))$$

We are, however, more interested in polynomial time functions in M. There are different ways to define ptime in a nonstandard model. One could just replace $\mathbb{N}$ by M everywhere in the definition above. This approach yields a class to big for our purposes as algorithms bounded by an nonstandard polynomial can be very powerful. Instead we are interested only in the extensions to M of the standard polynomial time functions.

Definition 2.16. We say that a definable M-function $f : \{0,1\}^* \to \{0,1\}^*$ is **polynomial time** or short **ptime** if there is a function symbol $\overline{f}$ in L_{all} such that $\mathbb{N} \models$ '$\overline{f}$ is ptime$'$ and $M \models \forall x \in \{0,1\}^* \ \overline{f}(x) = f(x)$ hold.

This definition can obviously not be formulated in M and this version of being ptime is, actually, not definable in M.

Example 2.4. Assume being ptime would be M-definable. Then the formula $\phi(k) :=$ '$|x| \to |x|^k$ is ptime' would define $\mathbb{N}$, contradicting the overspill.

We are considering polynomial time functions in M because we can use then to define an interesting family of random variables.

Definition 2.17. Let $\Omega \subseteq \{0,1\}^*$ be a sample space. Then we define **the family of ptime functions** F^{Ω}_{PV} by

$$F^{\Omega}_{PV} := \{\alpha : \Omega \to M \mid \alpha \text{ is the restriction of a ptime function } f : M \to M\}$$

.

F^{Ω}_{PV} is an example of a family of random variables that is closed under definition by cases by quantifier free L-formulas, at least if we choose the right L.

Lemma 2.17. *Assume all function symbols in $L \subseteq L_{all}$ are interpreted in M by ptime functions and all relation symbols in L are decidable by ptime functions in M. Then F^{Ω}_{PV} is an L-closed family of random variables that is closed under definition by cases by quantifier free L-formulas.*

Proof. It is trivial that F^{Ω}_{PV} is a family of random variables because every function in F^{Ω}_{PV} has domain Ω by definition and every ptime function f is definable in M using the function symbol $\overline{f}$ hence also its restriction to the definable set Ω.

We know that the polynomial time functions in $\mathbb{N}$ are closed under composition. For random variables $\alpha_1, \alpha_2 \ldots \alpha_k$ we find L_{all}-symbols $\overline{f}_1, \overline{f}_2, \ldots \overline{f}_k$ such that $M \models \forall x \in \Omega(\overline{f}_i(x) = \alpha_i(x))$ and $\mathbb{N} \models$ '$\overline{f}_i$ is ptime$'$ hold for all $i \in \{1, 2, \ldots k\}$. Now let $\overline{f} \in L$ be a function symbol. Then $\mathbb{N} \models$ '$\overline{f}(\overline{f}_1, \overline{f}_2, \ldots \overline{f}_k)$ is ptime$'$ hence there is a ptime function f in M such that $M \models \forall x \in \{0,1\}^*(f(x) = \overline{f}(\overline{f}_1(x), \overline{f}_2(x), \ldots \overline{f}_k(x))$ holds. The restriction of f to Ω is in F_{PV}^{Ω} therefore F_{PV}^{Ω} is L-closed.

It is also a well known fact that the ptime functions in $\mathbb{N}$ are closed under definition by cases by quantifier free L-formulas if all relations and functions in L are ptime computable. Therefore, F_{PV}^{Ω} is closed under definition by cases by quantifier free L-formulas in M by an analogous argument to the one above. □

Later, we want to use the biggest possible language such that the lemma above holds. This motivates the following definition.

Definition 2.18. Let $L_{PV} \subseteq L_{all}$ be the language consisting of function symbols for all ptime functions in $\mathbb{N}$ and the relation symbols $<$.

Obviously lemma 2.17 is applicable to this language hence F_{PV}^{Ω} is an L_{PV}-closed family of random variables that is closed under definition by cases by quantifier free L_{PV}-formulas.

On the other hand it is easy to see that F_{PV}^{Ω} is not closed under definition by cases even for a weak language like $L_{PA} := \{+, \times, 0, 1 <\}$. For example the halting set H is definable in this language. Therefore, if the ptime functions were closed under definition by cases

$$f(x) := \begin{cases} 1 & \text{if } x \in H \\ 0 & \text{else} \end{cases}$$

would define a ptime function because $0, 1$ are clearly ptime. However, this can not be the case as the halting problem is undecidable.

Chapter 3

An independence proof for bounded arithmetic

This chapter will be dedicated to showing an application of the method described in chapter 2. We build a boolean valued model of $Th_{\forall}(\mathbb{N})$ and show, under some complexity theoretic assumptions, that it doesn't model a specific form of the pigeon hole principle. This construction is taken from [33] and is arguably the easiest independence proof presented in the book. We begin by stating the theorem we intend to prove. This theorem has already been proven in [34] without reference to the forcing framework, using instead a famous result by Sam Buss[1].

Theorem 3.1. *Assume that there is a standard $\epsilon > 0$ such that no polynomial time algorithm can decrypt a message encrypted by a pair from RSA_k for more than a $(1 - \epsilon)$ fraction of pairs $(m^e \mod N, (e, N))$, where m is a message and (e, N) a RSA_k pair, for all k large enough. Then $Th_{\forall}(L_{PV})$ does not prove the weak pigeonhole principle for circuits.*

We have introduced neither RSA nor $Th_{\forall}(L_{PV})$ nor the weak pigeon hole principle for circuits. Hence, we will begin this chapter by giving a short introduction to the RSA method, the theory of circuits and the different forms of the pigeon hole principle. In all cases, we don't intend to give a comprehensive overview but focus on the topics that are necessary for a proper understanding of the theorem above and its proof. The introductions are mostly self contained, however not, all proofs are presented and a small amount of knowledge in algebra and complexity theory is assumed.

[1] Strictly speaking a slightly different security assumption is used.

3.1 Preliminaries

The definition of the theory $Th_\forall(L_{PV})$ is straight forward.

Definition 3.1. Let $L \subseteq L_{all}$ be a language, then $Th_\forall(L)$ is the collection of all universal L-sentences true in the restriction of the L_{all} model $\mathbb{N}$ to L.

3.1.1 The RSA method

Our presentation of the RSA system is taken mainly from [30], a book that can also be recommended as an introduction to cryptography in general. The RSA public key cryptosystem was developed by Rivest, Shamir and Adleman in 1977[2] [43] in responses to a question proposed by Diffie and Hellman [20].

The problem of secure communication over an insecure connection is generally modeled in cryptography by the following set-up. Two parties, usually called Alice and Bob, can exchange messages coded by strings, a third party, usually called Eve for eavesdropper, can read all these messages. In spite of this eavesdropping, Alice has to convey a secret message m to Bob without revealing it to Eve.

The RSA cryptosytem is a so called public key cryptosystem. It allows Alice and Bob to exchange a message in the following way: Bob publishes a public key that is known to Alice and Eve. Using this public key, Alice can encrypt her message in such a way that only Bob can decrypt it. This is achieved using (fairly simple) number theoretic methods. Therefore, we have to recall a few definitions and results from number theory. A more thorough treatments of these topics can be found in the already mentioned [30] or in any introductory textbook on number theory, for example [25].

Reminder 3.1. Let n be a natural number, a, b integers. We call a and b congruent modulo n, or in symbols $a \equiv b \mod n$, if $a - b$ is divisible by n. This defines, for a fixed n, an equivalence relation on the integers that is compatible with addition, multiplication and exponentiation.

The RSA method is based on two rather well known results. The first one is a special case of Euler's formula.

Proposition 3.2. *Let p and q be two distinct odd primes and k an integer coprime to pq (i.e $\gcd(k, pq) = 1$), then*

$$k^{(p-1)(q-1)/2} \equiv 1 \mod pq$$

[2] The British intelligence agency GCHQ had its own version of the RSA system since the early seventies but the inventors at GCHQ hadn't published their findings for security reasons. So it seems fair to attribute the RSA method to Rivest, Shamir and Adleman. More about the history of RSA can be found in [19].

holds.

We prove this theorem using Fermat's little theorem stating, for a prime p and an integer k, that $k^{p-1} \equiv 1 \mod p$ holds given that k is not divided by p. We refrain from presenting the proof of Fermat's little theorem. It can be found in [30] or [25].

Proof. By assumption we know that $q-1$ is divided by 2 therefore $\frac{q-1}{2}$ is an integer. Hence, we know that

$$k^{\frac{(p-1)(q-1)}{2}} = \left(k^{(p-1)}\right)^{\frac{(q-1)}{2}}$$

holds. Since k is not divided by p by assumption we can apply Fermat's little theorem to get

$$\left(k^{(p-1)}\right)^{\frac{(q-1)}{2}} \equiv 1^{\frac{(q-1)}{2}} \equiv 1 \mod p$$

Using the same argument with the roles of p and q switched we get

$$k^{\frac{(p-1)(q-1)}{2}} \equiv 1 \mod q$$

Therefore, $k^{\frac{(p-1)(q-1)}{2}} - 1$ is divisible by p and q so also by pq or in other words

$$k^{\frac{(p-1)(q-1)}{2}} \equiv 1 \mod pq$$

□

The second result is proven using the Extended Euclidean Algorithm. We will treat this algorithm as a black box. It is thoroughly described in section 1.3 in [25] and can be found as theorem 1.11 in [30].

Proposition 3.3. *Let p and q be two distinct primes and e a natural number coprime to $(p-1)(q-1)$. Then e has a multiplicative inverse d modulo $(p-1)(q-1)$, i.e $ed \equiv 1 \mod (p-1)(q-1)$ holds. Furthermore, the congruence $x^e \equiv c \mod pq$ has the solution $x \equiv c^d \mod pq$.*

Proof (Sketch). The Extended Euclidean Algorithm finds for any two natural numbers a and b two integers u and v such that $au + bv = \gcd(a, b)$ holds. Hence, using this algorithm, we find u, v such that $eu + (p-1)(q-1)v = 1$ holds. This yields $eu - 1 = -(p-1)(q-1)v$ so by definition $eu \equiv 1 \mod (p-1)(q-1)$.

Assume m, and hence c is coprime to N. Then proving that c^d is a solution to $x^e \equiv c \mod pq$ is done by a short computation. The congruence $de \equiv 1 \mod (p-1)(q-1)$ implies that there is a $k \in \mathbb{Z}$ such that $de = 1 + k(p-1)(q-1)$ holds. Using this we get

$$\begin{aligned}
(c^d)^e &\equiv c^{de} \mod pq \\
&\equiv c^{1+k(p-1)(q-1)} \mod pq \\
&\equiv c\big(c^{(p-1)(q-1)}\big)^k \mod pq \\
&\equiv c1^k \mod pq \\
&\equiv c \mod pq
\end{aligned}$$

where the step from line three to four uses Euler's formula.

For the (in practice basically irrelevant) special case that m is not coprime to N a similar (but longer) proof using Fermat's little theorem can be used to prove the same result. □

Using these two propositions, we can now describe the RSA algorithm. The algorithm is divided into the following three steps:

- **Key creation** Bob picks two secret primes p and q and an encryption exponent e that is coprime to $(p-1)(q-1)$. He then computes $pq = N$ and sends his public key, the tuple (e, N), to Alice whereby it will become known to Eve, too.
- **Encryption** Alices encrypts the message $m < N$ using Bobs public key by computing $c \equiv m^e \mod N$, the so called ciphertext. Subsequently, she sends c to Bob (and therefore also to Eve).
- **Decryption** Given, p, q, e and c Bob computes the message m. First he computes d the inverse of e modulo $(p-1)(q-1)$, which exists by proposition 3.3. This can be done in polynomial time using the Extended Euclidean Algorithm. Then he computes $c^d \mod N$ which is equivalent to m by proposition 3.3.

The RSA method yields a secure way of communication for Alica and Bob if it is not feasible to compute m from e, N and c. In general, there is no method know that is quicker than factoring N to obtain p and q and then compute m the same way Bob does.[3] With the current state of knowledge this is not feasible for large primes p and q. However, there isn't much known about the complexity of these problems.[4]

We know that factoring is a element of $NP \cap coNP$ [8] which is assumed to be a proper subset of NP but also a proper superset of P. Assuming

[3] For certain choices of p, q and e faster ways of computing m are known. Therefore, Bob has to be careful in his choice of these values, see, for example, [30].

[4] More precisely about the complexity of the corresponding decision problems.

Bob	**Alice**
Key creation	
Choose distinct primes p and q and a natural number e coprime to $(p-1)(q-1)$. Compute $N = pq$ and publish (e, N).	
	Encryption
	To encrypt a message m compute $c \equiv m^e \mod N$. Send c to Bob.
Decryption	
Find a d such that $ed \equiv 1 \mod (p-1)(q-1)$. Then compute $c^d \equiv m \mod N$.	

Table 3.1: The RSA cryptosystem

$coNP \neq NP$, this means factoring is neither NP nor $coNP$ complete. It is also widely believed that factoring is not in P making it a candidate for a problem that is complete for a intermediate class of problems between P and NP. Moreover, the so called RSA problem of computing m from e, N and c may or may not be as hard as factoring. There is no strong consensus on this question among researchers [9].

Now, we collect possible public keys into sets of so-called RSA-pairs.

Definition 3.2. We call a pair (e, N) of natural numbers an RSA_k**-pair** if

- N is the product of two distinct primes p and q of length k (in binary) (and therefore $|N| = 2k$).
- $1 < e < N$ holds and e is coprime to $(p-1)(q-1)$.

Furthermore, for a natural number k **the set** RSA_k is defined as the collection of all RSA_k-pairs.

The assumption that p and q have the same length is not necessary for the proof of theorem 3.1, however in practice this is often the case as this ensures the best compromise between security and speed because in general the difficulty of factoring depends on the smallest prime factor. Therfore this assumption guarantees that we cover an significant fraction of RSA implementations as they are used in practice. However, the proof works the same without this assumption.

Finally, we cover an alternative way to break an RSA encryption that will be used in the proof. This approach is discussed for example in [34].

Lemma 3.4. *Assume (e, N) is a RSA_k pair for some k and c is a message encrypted with (e, N). Then if we can find an integer w such that $c^w \equiv 1 \mod N$ holds, we can decrypt c.*

Proof. First, we check if c is a divisor of N. If this is the case, we are done because we have factored N as it has only two factors. Therefore, we can assume that c is coprime to N.

Assume we have found a w such that $c^w \equiv 1 \mod N$ holds. Let r denote the order of c modulo N, i.e the smallest k such that $c^k \equiv 1 \mod N$ holds. Then Euler's formula, which is applicable because c and N are coprime, implies that r divides $(p-1)(q-1)$. Then, e and r are coprime because e and $(p-1)(q-1)$ are coprime. Furthermore, r divides w. Now we define $w_0 := \frac{w}{\gcd(w,e)}$. Then, w_0 and e are coprime but r divides w_0. This implies

$$c^{w_0} \equiv c^{\frac{w_0}{r} r} \equiv 1^{\frac{w_0}{r}} \equiv 1 \mod N$$

Moreover, when d is the secrete key, the following holds:

$$m^{w_0} \equiv (c^d)^{w_0} \equiv (c^{w_0})^d \equiv 1 \mod N$$

Now, using the Extended Euclidean Algorithm we can find the inverse d' of e modulo w_0 because e and w_0 are coprime by definition. Hence, the following holds for some k:

$$c^{d'} \equiv (m^e)^{d'} \equiv m^{kw_0+1} \equiv m \mod N$$

Therefore, we have decrypted c. □

3.1.2 Boolean circuits

In this section we introduce boolean circuits. These are covered more extensively for example in chapter 6 of [5] or in [47]. The following presentation is based on these two books.

Circuits have been studied since the 1940s with the first important results on boolean circuits published in 1949. The use of boolean circuits in complexity theory started in the late 1970s. Unfortunately, progress on the big open questions of circuit complexity got stuck in the 1990s. Nevertheless, boolean circuits are still an important tool in the theoretical computer sciences today.

Definition 3.3. A **boolean circuit C with n-input and m-output** is a directed acyclic labeled graph with the following properties:

- C has n sources (vertices with fan-in 0) bijectively labeled with in_1 to in_n.
- All nonsources vertices are labeled by $\wedge$, $\vee$ or $\neg$[5].
- The graph has m sinks (vertices with fan-out 0) additionally bijectively labeled by out_1 to out_m.
- Every vertex with label $\neg$ must have fan-in 1.

We call the vertices of C also its **gates**. The output of C on input $x \in \{0,1\}^n$ is defined by a recursive valuation of C:

- The source with the label in_i is assigned the value corresponding to the i-th bit of x.
- A gate labeled by $\wedge$ is assigned the value 1 if all its predecessor have value 1 and value 0 otherwise.
- A gate labeled $\vee$ is assigned value 1 if one of its predecessors has value 1 and 0 otherwise.
- A gate labeled by $\neg$ is assigned the opposite value of its predecessor.
- The value of the j-th bit of **the output** is defined by the value of gate out_j.

We write $C(x)$ for the output of the circuit C on input x. Finally we say a function f is **computed by a circuit** C with n-input if the domain of f is $\{0,1\}^n$ and $C(x) = f(x)$ holds for all $x \in \{0,1\}^n$.

It is not hard to see that the valuation of C is well defined. Furthermore, functions computed by circuits have obviously domain $\{0,1\}^n$ for some n and range $\{0,1\}^m$ for some m. However, the following theorem shows that this is the only restriction on the computational power of circuits.

Lemma 3.5. *Every function $f : \{0,1\}^n \to \{0,1\}^m$ with $n, m \in \mathbb{N}$ can be computed by a circuit.*

Proof (sketch). It is easy to see that we can construct for every string $x \in \{0,1\}^n$ a circuit C_x that outputs 1 on input x and 0 otherwise (see figure 3.1). Now we construct a big circuit C_f by combining circuits C_x for every $x \in \{0,1\}^n$ in the following way: We label every output gate[6] v_i $(i = 1, \ldots m)$

[5] As usual the choice of logical symbols is somewhat arbitrary as long as we have enough expressiveness.

[6] i.e gate labeled with out_i for some i

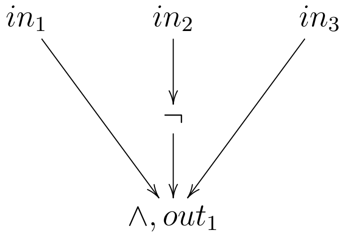

Figure 3.1: A circuit accepting 101

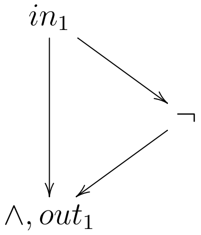

Figure 3.2: The circuit $C_\perp$ mapping any input to 0

with $\vee$ and connect each C_x to each output gate v_i either directly if the i-th bit of $f(x)$ is 1 or via a circuit $C_\perp$ (see figure 3.2) that maps 0 and 1 to 0 if the i-th bit of $f(x)$ is 0 (see figure 3.3). It is straightforward to check that C_f computes f. □

Using sequences of circuits, we can also compute arbitrary functions from strings to strings as long as they are length-respecting.

Definition 3.4. A function $f : \{0,1\}^* \to \{0,1\}^*$ is called **length-respecting**

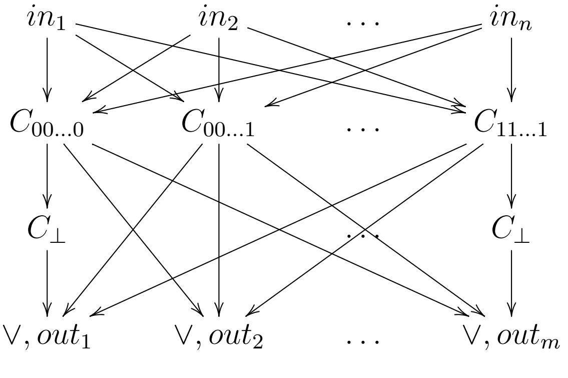

Figure 3.3: Sketch of C_f

if $|x| = |y|$ implies $|f(x)| = |f(y)|$.

Corollary 3.6. *For every length-respecting function $f : \{0,1\}^* \to \{0,1\}^*$ there is a sequence of circuits $(C_k)_{k\in\mathbb{N}}$ such that $f(x) = C_k(x)$ holds for all $x \in \{0,1\}^k$.*

Proof. Clear. □

Yet, it may be impossible to find these circuits in a computable fashion. For reasonable functions, however, we can compute such circuits in a similarly reasonable time. To be able to 'compute a circuit' we have to code circuits as strings. It is possible to do this in a way that a ptime function can evaluate a circuit based on its code.

Observation 3.1. We can code a circuit C in the following way: We enumerate all gates in C such that the input and output gates are ordered according to their labels (i.e for $i < j$ the gate labeled in_i is enumerated before the gate with the label in_j) and each gate is assigned a number higher than the numbers of all its predecessor. This is possible because a boolean circuit is acyclic. Furthermore, we fix an enumeration of the set $\{in, \wedge, \vee, \neg, (out, \wedge), (out, \vee), (out, \neg)\}$. The code for a gate is a tuple consisting of the number of the gate, the number of its label (ignoring the subscript for in and out labels) and the numbers of all its predecessors. The code of C is the set[7] consisting of the codes for all its gates.

It is easy to see that there is a ptime function $Circ(x)$ that decides if a set of tuples actually codes a boolean circuit. One only needs to check the conditions on the fan-in and fan-out of gates labeled by in_i, out_i or $\neg$ and the conditions posed in the coding. All these conditions can be checked for each gate with a ptime function. The condition that each gate is represented by a number higher than the numbers of all his predecessors guarantees the acyclicity of the graph. Furthermore, obviously the size of the in- and output can be easily read off using ptime functions $In(x)$ and $Out(x)$. Finally we can valuate a circuit on a string x using a ptime function $Val(C, x)$. The gates are evaluated in order of their position in the enumeration. By construction all values necessary to compute the value of a gate have already been computed in a previous step. Computing the value of a gate from the values of its predecessors is obviously ptime. In the end, the output can be read off the *out* labeled vetices according to their order in the enumeration.

[7] We have seen in reminder 1.2 that we can code sets into natural numbers. With more refined methods this can also be done in a way that these codes are can be read of in ptime.

Lemma 3.7. *Let f be a polynomial time computable, length-respecting function from $\{0,1\}^*$ to $\{0,1\}^*$. Then there is a polynomial time function g that computes on input 1^n a circuit C_n such that $C_n(x) = f(x)$ holds for all $x \in \{0,1\}^n$.*

A complete formal proof for this statement would necessarily be quite long, technical and not very interesting or difficult so we will only present a proof sketch. A few more details can be found in [5], a complete proof in [47].

Proof (sketch). f is polynomial time computable, therefore we know there is a turing machine $\mathbb{A}$ computing f in polynomial time. We can simulate every step of this machine with a boolean circuit because we can code the state of $\mathbb{A}$ at any given time by a string and, as $\mathbb{A}$ is a ptime machine, the length of this code can be polynomial in n. Furthermore, we can compute the next state of $\mathbb{A}$ from the current state using the transition function of $\mathbb{A}$. Lemma 3.5 guarantees that we can do this computation also using a boolean circuit. Particularly, this can be done using a circuit of polynomial size and this circuit can be generated in polynomial time. This fact can be proven by constructing a simulation of this function by a circuit and check that this simulation has the required properties. We skip this step because it depends heavily on the technical details of the turing machine in question.

Now let $p(n) > t_{\mathbb{A}}$ be a polynomial bounding the running time of $\mathbb{A}$. Then we can define a function $g(n)$ that constructs a circuit that simulates $\mathbb{A}$ for $p(n)$ steps and reads off the output from the final state. (To prove the feasibility of the last step one can again write down an actual boolean circuit accomplishing the step). This function is obviously ptime as it performs a ptime operation $p(n)$-times. □

3.1.3 The pigeonhole principle

The pigeonhole principle states, roughly speaking, that, if there are more pigeons than pigeonholes, at least two pigeons have to share a hole. More formally, there is no injective map from m to n for $n, m \in \mathbb{N}$ if $m > n$ holds. This principle is widely believed to have been used for the first time in a mathematical proof by the famous German mathematician Peter Gustav Lejeune Dirichlet in [22] and [21] in 1842[8]. To his honor the principle is often also called the "Schubfachprinzip", the name Dirichlet himself gave the principle, or the "Dirichletsche Schubfachprinzip (see [42]).

[8] Recent research suggest that this believe is wrong and the principle has been used much earlier. See [42].

This principle is obviously true, the proof complexity of a propositional form of this principle is , however, quite high in many important proof systems[9]. There has been extensive research on the proof complexity of the pigeonhole principle and its variations. In general, the proof complexity of the PHP depends on the relation of the parameters m and n where the hardest cases[10] is $m = n + 1$. The case we are interested in, sometimes called the weak pigeonhole principle or WPHP, is $m = 2n$, which has a lower proof complexity in some proof systems. A survey of the proof complexity of the pigeonhole principle for different ratios of m and n can be found in [40]. Finally we can adapted the pigeonhole principle to circuits by talking about the map computed by a circuit C. This allows us to formulate the weak pigeonhole principle for circuits. We are interested in the formulation of the weak pigeonhole principle in L_{PV}. To that end let $Circ(x), In(x), Out(x)$ and $Val(C, x)$ be the functions defined in observation 3.1.

Definition 3.5. The **weak pigeonhole principle for circuits** is the L_{PV}-sentence $\forall C \forall a WPHP(C, a)$ where $WPHP(C, a)$ is the formula

$$\begin{aligned}&\neg(Circ(C) = 1 \wedge In(C) = |a| + 1 \wedge Out(C) = |a|)\\ &\qquad \vee \exists (u, v) \in \{0,1\}^{|a|+1} \times \{0,1\}^{|a|+1} (u \neq v \wedge Val(C, u) = Val(C, v))\end{aligned}$$

stating "either C is not a circuit computing a map from $\{0,1\}^{|a|+1}$ into $\{0,1\}^{|a|}$ or there are $u \neq v \in \{0,1\}^{|a|+1}$ such that $C(u) = C(v)$ holds".

3.2 The proof

Before presenting its proof we recall the statement of theorem 3.1.

Theorem 3.1. *Assume that there is a standard $\epsilon > 0$ such that no polynomial time algorithm can decrypt a message encrypted by a pair from RSA_k for more than a $(1 - \epsilon)$ fraction of pairs $(m^e \mod N, (e, N))$, where m is a message and (e, N) a RSA_k pair, for all k large enough. Then $Th_\forall(L_{PV})$ does not prove the sentence the weak pigeonhole principle for circuits.*

Two final remarks on this statement: First, the assumption on the security of RSA is quite mild, certainly it wouldn't suffice to guarantee the security of RSA in our daily life. Secondly, the proof in [33] is not entirely correct. However, it can easily be fixed using an idea from the proof of a similar statement in [34].

[9] For some interesting philosophical thoughts on this surprising result one can consult [1].

[10] Sometimes only this case is called the pigeonhole principle.

Proof. We assume there is a standard $\epsilon > 0$ and a $k_0 \in M$ such that no polynomial time algorithm can break more than a $(1-\epsilon)$ fraction of RSA encryptions based on a pair from RSA_k for all $k > k_0$, i.e there is no ptime function that decrypts the cypertext c for more than a $(1-\epsilon)$ fraction of pairs $(c(e,N))$ where (e,N) is a RSA_k pair and $c < N$ holds.

We build a model $K(F)$ as defined in definition 2.11. Let n be an arbitrary nonstandard element of M bigger than k_0. We take as the sample space

$$\Omega = \{(c,(e,N)) \mid \exists m \in \{0,1\}^*(c \equiv m^e \mod N), c < N, (e,N) \in RSA_{\frac{n}{2}}\}$$

As a family of random variables over the sample space Ω we take F_{PV}^{Ω} as in definition 2.17. Finally as a language we pick L_{PV} as in definition 2.18. By lemma 2.17 we know that F_{PV}^{Ω} is L_{PV} closed, therefore the model $K(F_{PV}^{\Omega})$ is defined and, by lemma 2.10, a model of $Th_{\forall}(L_{PV})$. We assume for the sake of a contradiction that

$$K(F_{PV}^{\Omega}) \vDash \forall C \forall a WPHP(C,a)$$

holds.

Let $\alpha : \Omega \to M$ be a function in M that computes from $(c,(e,N))$ a circuit $C_{c,e,N}$ that computes $x \in \{0,1\}^{n+1} \to c^x \ mod\ N \in \{0,1\}^n$ (to achieve $c^x \mod N \in \{0,1\}^n$ we can assume that the output has an appropriate amount of leading zeros). Note that α is the restriction of a ptime function to Ω because of lemma 3.7 and the fact that modular exponentiation is ptime. Therefore, α is an element of F_{PV}^{Ω}. Hence, we know that

$$K(F_{PV}^{\Omega}) \models WPHP(\alpha, 2^n)$$

is true by assumption. Observe that this statement makes sense because the projected length function $(x,(y,z)) \to |z|$ is ptime and constantly n in $K(F_{PV}^{\Omega})$ hence $WPHP(\alpha, 2^n)$ can be formulated in the model. Moreover, we know that

$$[\![\neg(Circ(\alpha) = 1 \wedge In(\alpha) = |n| + 1 \wedge Out(\alpha) = |n|]\!] = 1_{\mathcal{B}}$$

is true because $\alpha((c,(e,N)))$ is a circuit computing a map from $\{0,1\}^{|n|+1}$ into $\{0,1\}^{|n|}$ in M for all $(c,(e,N)) \in \Omega$. Combining these two facts we can conclude

$$[\![\exists(u,v) \in \{0,1\}^{|n|+1} \times \{0,1\}^{|n|+1}(u \neq v \wedge Val(\alpha,u) = Val(\alpha,v))]\!] = 1_{\mathcal{B}}$$

By lemma 2.17 F_{PV}^{Ω} is closed under definition by cases by open L_{PV}-formulas. Therefore, by the witnessing theorem 2.16 there is a $\beta \in F_{PV}^{\Omega}$ such that

$$\mu([\![\beta = (u,v) \in \{0,1\}^{|n|+1} \times \{0,1\}^{|n|+1} \wedge u \neq v \wedge Val(\alpha,u) = Val(\alpha,v))]\!]) \geq 1 - \frac{\epsilon}{3}$$

holds. Using lemma 2.12 and the definition of α we know that

$$\begin{aligned}\underset{(c,(e,N))\in\Omega}{Prob}[\beta(c,(e,N)) &= (u,v)\in\{0,1\}^{|n|+1}\times\{0,1\}^{|n|+1}\\ &\wedge\ u\neq v\wedge c^u=c^v\ mod\ N]\geq 1-\frac{\epsilon}{2}\end{aligned}$$

holds in M. In other words β finds for a $(1-\frac{\epsilon}{2})$ fraction of pairs $(c,(e,N))\in\Omega$ a pair (u,v) such that $c^u=c^v\ mod\ N$ holds. But then we can in polynomial time compute $w=u-v$ which has the property that $c^w=1\ mod\ N$ holds and by lemma 3.4 this is enough decrypt the cyphertext c. Therefore, our assumption about the security of RSA is wrong in M. As M is a model of true arithmetic this means it is also wrong in $\mathbb{N}$ contradicting our assumption. Therefore,

$$K(F^{\Omega}_{PV})\not\models\forall C\forall aWPHP(C,a)$$

must be true, hence we have constructed a model of $Th_{\forall}(\mathbb{N})$ that doesn't model the weak pigeonhole principle for circuits. □

Observation 3.2. It is worth remarking that this proof gives us independence from an important theory of bounded arithmetic without further work. Namely, the theory PV mentioned in the introduction is a universal theory in the language L_{PV}. Therefore, a model of $Th_{\forall}(\mathbb{N})$ in the language L_{PV} is obviously a model of PV, hence we have proven the independence of the weak pigeonhole principle for circuits from PV. Moreover, we can also easily get independence form another theory mentioned in the introduction, S^1_2, using a well known but deep result from bounded arithmetic: $WPHP(C,a)$ is a $\forall\exists$-sentence and S^1_2 is $\forall\exists$-conservative over PV (see [12]), hence the weak pigeonhole principle for circuits is also independent from S^1_2.

Chapter 4

Hard sequences, derandomization and pseudo proof systems

In section 24.4 of [33] the concept of a pseudo proof system is introduced, but not thoroughly developed. In this chapter we present the definition of a pseudo proof system and the closely related concept of approximate p-simulations. Moreover, we present a few new observations regarding these concepts.

4.1 Proof systems and p-simulations

Propositional proof systems in the sense of Cook-Reckhow are the key concept for proof complexity and play an important role in complexity theory in general. They were introduced in [18] as a generalization of the classical concept of a proof system. In order to talk about proof systems in a complexity theoretic setting we have to fix an encoding of formulas.

Notation 4.1. We fix a reasonable[1] encoding of formulas and assignments for these formulas as binary strings with the property that any possible assignment for a formula ϕ has a code that is shorter than the code of the formula. Furthermore, we assume that every string codes an assignment for every formula. This is possible if we interpret the first k bits of a string as the truth values of the first k distinct variables. If there are more bits than variables, we ignore the extra bits. If there are less, we set the extra variables

[1] In the sense that we can perform operations as negating a formula in ptime.

to 0. In the following we will treat these codes as if they were the actual formulas.

Using these codes we can define proof systems from a complexity theoretic perspective.

Definition 4.1. We call a polynomial time function P a **propositional proof system** if its range is exactly the set of all tautologies. We call a string $x \in \{0,1\}^*$ a **P-proof** of a formula α if $P(x) = \alpha$.

Cook and Reckhow simultaneously introduced in the same paper the important notion of a p-simulation.

Definition 4.2. Let P and Q be propositional proof systems. Then P **p-simulates** Q if there exists a ptime function f, called a **p-simulation of** Q **in** P, such that $P(f(x)) = Q(x)$ holds for all $x \in \{0,1\}^*$.

There are many important open questions regarding propositional proof systems that are closely linked to more general open questions in complexity theory. For example, it is open if a p-optimal proof system exists, i.e. a propositional proof system that p-simulates all other propositional proof systems. The non-existence of a p-optimal proof system would imply $P \neq NP$ (see [32]). Another important open question is whether a propositional proof system exists such that all tautologies have a proof which size is polynomially bounded in the size of the tautology. The existence of such a proof system is equivalent to $NP = coNP$ ([18]).

4.2 Pseudo proof systems and approximate p-simulations

In the following, we work in models of the form $K(F_{PV}^{\{0,1\}^n})$, similar to the one used to prove theorem 3.1. For a better readability we use the notation $K(F_{PV}^{\{0,1\}^n}) =: K(F_{PV}^n)$. Initially we work in the model $K(F_{PV}^n)$ for a fixed arbitrary nonstandard n. Every polynomial time function in M is also represented by an element of $K(F_{PV}^n)$, therefore we can ask how a function that is a propositional proof system in M behaves in $K(F_{PV}^n)$. As we will see, any propositional proof system will look like a tautology to $K(F_{PV}^n)$. To make this a precise statement we have to make a few technical definitions.

Definition 4.3. Let $\mathrm{Sat}(x,y)$ be the L_{PV}-symbol for a ptime relation that decides if x is the code for a formula and if y is a satisfying assignment for the formula x. Then define the bounded universal L_{PV}-formula **Taut(x)** as $\mathrm{Taut}(x) := \forall y(|y| \leq |x| \rightarrow \mathrm{Sat}(x,y))$.

A polynomial time function of M is represented in $K(F_{PV}^n)$ only by its restriction to $\{0,1\}^n$. The following definition gives us a useful notation for this restriction.

Definition 4.4. Let P be a propositional proof system. We write $\alpha_P \in F_{PV}$ for the restriction of P to $\{0,1\}^n$ and call this the **$K(F_{PV}^n)$-representation** of Q

This gives us the statement alluded to above.

Proposition 4.1. *Let P be a propositional proof system and α_P its $K(F_{PV}^n)$-representation. Then $[\![\mathit{Taut}(\alpha_P)]\!] = 1_{\mathcal{B}}$ holds with respect to $K(F_{PV}^n)$.*

Proof. The following holds by definition of Taut(x) and $K(F_{PV}^n)$

$$[\![\mathrm{Taut}(\alpha_P)]\!] = \bigwedge_{\beta} [\![|\beta| \leq |\alpha_P| \to \mathrm{Sat}(\alpha_P, \beta)]\!]$$

Moreover, for every function $\beta \in F_{PV}^n$ we know

$$\underset{\omega \in \{0,1\}^n}{Prob} [|\beta(\omega)| \leq |\alpha_P(\omega)| \to \mathrm{Sat}(\alpha_P(\omega), \beta(\omega)] = 1$$

as the image of α_Q is always a tautology by definition. Hence, $[\![\mathrm{Taut}(\alpha_P)]\!] = 1_{\mathcal{B}}$ holds. □

In other words, proof systems in M (or more exactly their restrictions to the sample space Ω) are tautologies in $K(F_{PV}^n)$. However, a function α may be a tautology in $K(F_{PV}^n)$ without being the restriction of a proof system in M. This motivates the following definition.

Definition 4.5. We call $\alpha \in F_{PV}^n$ a **pseudo proof system** if $[\![\mathrm{Taut}(\alpha)]\!] = 1_{\mathcal{B}}$ holds with respect to $K(F_{PV}^n)$.

It is easy to see that there are pseudo proof systems that are not restrictions of proof systems in M.

Example 4.1. Take (the restriction of) a proper propositional proof system α_P and for every k change its output on 0^k to a formula that is not a tautology. Clearly this can be done in a way such that the resulting function is still ptime. The new function α'_P satisfies $[\![\alpha_P = \alpha'_P]\!] = 1_{\mathcal{B}}$ by definition and hence $[\![\mathrm{Taut}(\alpha'_P)]\!] = 1_{\mathcal{B}}$. However, as $\alpha'_P(0^k)$ is no tautology α'_P can not be the restriction of a proof system to $\{0,1\}^n$.

Generally it is quite easy to construct pseudo proof systems that err on an infinitesimal fraction of inputs. However, it could be possible that a pseudo proof system never outputs a tautology if no function in F_{PV}^n can find the falsifying assignments for these outputs often enough. We will discuss this possibility in the next section.

Prior to that we want to generalize the notion of a p-simulation to so-called approximate p-simulations. This is not as straightforward as one would expect. A pseudo proof system α_P is an element of $K(F_{PV}^n)$ and not a function, therefore there is no obvious meaningful way to talk about composing α_P with a simulation f in $K(F_{PV}^n)$. However, $K(F_{PV}^n)$ is a L_{PV} model and there is a symbol for P in L_{PV}. We can utilize this fact for our definition.

Definition 4.6. Let P be a propositional proof systems in M and let α be a pseudo proof system in $K(F_{PV}^n)$. If $[\![\exists z P(z) = \alpha]\!] = 1_{\mathcal{B}}$ holds we say P **approximately p-simulates** α. If $\alpha = \alpha_Q$ holds, where α_Q represents a propositional proof system Q in $K(F_{PV}^n)$, we say P **approximately p-simulates** Q **on** $\{0,1\}^n$.

Note that it is not clear how this definition could be extended to allow approximate p-simulation by a pseudo proof system, as there is not necessarily a unique symbol representing a pseudo proof system in M.

Calling this notion approximate p-simulation can be justified by the following observation.

Observation 4.1. Let P be a propositional proof systems in M, let α be a pseudo proof system in $K(F_{PV}^n)$ and assume P approximately p-simulates α. Then, by the witnessing theorem 2.16, for every standard $\epsilon > 0$ there is a $\gamma \in F_{PV}^n$ such that

$$\mathrm{Prob}_{\omega \in \{0,1\}^n}[P(\gamma(\omega)) = \alpha(\omega)] > 1 - \epsilon$$

holds.

Mimicking the open question about an optimal propositional proof system we could ask if a propositional proof system exists that is optimal with regard to approximate p-simulations and pseudo proof systems, i.e. it approximately p-simulates all pseudo proof systems, either in one, or even in all models $K(F_{PV}^n)$. We will show in the next section that, under some complexity theoretic assumptions, there is a model $K(F_{PV}^n)$ such that no proof system is optimal with regard to approximate p-simulations and pseudo proof systems in this model. This will be done by constructing a pseudo proof system that errs everywhere.

4.3 Pseudo proof systems that err everywhere

How far away from being a propositional proof system can a pseudo proof system be? The worst case scenario for a pseudo proof system would be that it never outputs a tautology. This motivates the following definition.

Definition 4.7. Fix a $n \in M$. A pseudo proof system α **errs everywhere** if, for every $\omega \in \{0,1\}^n$, it holds that $\alpha(\omega)$ is not a tautology in M.

Do these worst case pseudo proof systems exist? It turns out that this question is equivalent to asking if certain hard sequences exist.

4.3.1 Hard Sequences

A introduction to hard sequences can be found in [15]. All classical definitions and results in this section are taken from [15] if not attributed otherwise. The definition for an "ordinary" hard sequences is the following:

Definition 4.8. A sequence $(x_s)_{s\in\mathbb{N}}$ of strings $x_s \in \{0,1\}^*$ is called a **hard sequence** for an algorithm $\mathbb{A}$ deciding a problem Q if the following holds

- $x_s \in Q$ for all $s \in \mathbb{N}$.
- $1^s \rightarrow x_s$ is ptime computable.
- $t_{\mathbb{A}}(x_s)$ is not polynomially bounded in s

A hard sequence is called **strongly hard** if additionally $s < |x_s|$ holds for all $s \in \mathbb{N}$.

We are interested in hard sequences for SAT-solvers. Therefore, we have to define hard sequences for search problems. Search problems are defined as follows:

Definition 4.9. A **search problem** $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is a set of pairs of strings. An algorithm $\mathbb{A}$ solves R if

- the algorithm $\mathbb{A}$ accepts x with output z such that $(x,z) \in R$ holds if there is a y such that $(x,y) \in R$ holds
- $\mathbb{A}$ rejects otherwise.

There are two possible definitions of a hard sequence for a search problem. We present only one and explain the reason we choose this one later.

Definition 4.10. A sequence $(x_s)_{s\in\mathbb{N}}$ of strings $x_s \in \{0,1\}^*$ is called a **hard sequence** for an algorithm $\mathbb{A}$ solving a search problem R if the following holds

- for all $s \in \mathbb{N}$ there is a y_s such that $(x_s, y_s) \in R$ holds.
- $1^s \to x_s$ is ptime computable.
- $t_{\mathbb{A}}(x_s)$ is not polynomially bounded in s

A hard sequence is called **strongly hard** if additionally $s < |x_s|$ holds for all $s \in \mathbb{N}$.

The alternative definition is a sequence of elements of R. This would be called a solved sequence.

We can now define the search problem we are interested in. The problem of solving SAT.

Definition 4.11. The search problem SAT consists of all pairs (ϕ, x) where ϕ is (the code of) a satisfiable formula and x is (the code of) a satisfying assignment of ϕ. An algorithm is called a SAT**-solver** if it solves SAT.

An important result about SAT-solver is due to Levin, who proved that there exists an almost optimal SAT-solver [36].

Definition 4.12. A SAT-solver $\mathbb{A}^*$ is **almost optimal** if for every SAT-solver $\mathbb{B}$ there is a polynomial p s.t. for all ϕ satisfiable $t_{\mathbb{A}}(\phi) < p(t_{\mathbb{B}}(\phi) + |\phi|)$ holds.

The proof of Levin's theorem is a clever application of diagonalization, heavily using the properties of search problems. Indeed the same result holds for all NP-search problems.

Proposition 4.2 (Levin)**.** *There is an almost optimal SAT-solver.*

Proof (Sketch). Let $(\mathbb{A}_k)_{k\in\mathbb{N}}$ be an enumeration of all possible algorithms and let $\mathbb{B}$ be a SAT-solver. We define an algorithm $\mathbb{A}^*$ in the following way. In the first iteration $\mathbb{A}^*$ simulates $\mathbb{A}_1$ and $\mathbb{B}$ for one step. In the second iteration it simulates $\mathbb{A}_1$ and $\mathbb{B}$ for two steps and $\mathbb{A}_2$ for one step. With each following iteration $\mathbb{A}^*$ doubles the steps of each simulation and adds one $\mathbb{A}_i$. Every time one of the $\mathbb{A}_i$ or $\mathbb{B}$ stops, $\mathbb{A}^*$ checks if the output is a satisfying assignment for the input x. If it is a satisfying assignment $\mathbb{A}^*$ stops and outputs the assignment. If $\mathbb{B}$ rejects, so does $\mathbb{A}^*$. Otherwise $\mathbb{A}^*$ keeps on running. As $\mathbb{B}$ is a SAT-solver, so is $\mathbb{A}^*$.

Now, let $\mathbb{C}$ be a SAT-solver. By definition there is a $k^* \in \mathbb{N}$ such that $\mathbb{C} = \mathbb{A}_{k^*}$ holds. Therefore, the running time of $\mathbb{A}^*$ is polynomially bounded in the running time of $\mathbb{C}$, as simulating another algorithm yields only a polynomially slowdown. □

For decision problems almost optimal algorithms cannot have hard sequences [15] but this is not true for search problems such as solving SAT if we use the definition of a hard sequence from above instead of solved hard sequences. This also reduces the question if there are hard sequences for all SAT-solver to the question if there are hard sequences for the almost optimal SAT-solver as one easily sees

Proposition 4.3. *If $(x_s)_{s\in\mathbb{N}}$ is hard for the almost optimal SAT-solver $\mathbb{A}^*$, then it is hard for every SAT-solver.*

Proof. Let $\mathbb{B}$ be a SAT-solver, $(x_s)_{s\in\mathbb{N}}$ a hard sequence for $\mathbb{A}^*$. Assume $(x_s)_{s\in\mathbb{N}}$ is not a hard sequence for $\mathbb{B}$. Then there is a polynomial p such that $t_{\mathbb{B}}(x_s) < p(s)$ holds. But there is also a monotone polynomial p' such that $t_{\mathbb{A}^*}(x_s) < p'(t_{\mathbb{B}}(x_s) + |x_s|)$ holds, which implies $t_{\mathbb{A}^*}(x_s) < p'(p(s) + |x_s|)$ and, as $|x_s|$ is polynomially bounded in s by definition, this contradicts the assumption that $(x_s)_{s\in\mathbb{N}}$ is hard for $\mathbb{A}^*$. □

From this we get easily the following important corollary.

Corollary 4.4. *If $(x_s)_{s\in\mathbb{N}}$ is hard for the almost optimal SAT-solver $\mathbb{A}^*$, then there is no $s_0 \in \mathbb{N}$ and no ptime algorithm $\mathbb{B}$ such that $\mathbb{B}$ solves all x_s, i.e finds a satisfying assignment for x_s, for $s > s_0$.*

Proof. Assume otherwise there is a ptime algorithm $\mathbb{B}$ solving x_s for all s bigger than some $s_0 \in \mathbb{N}$. Let $\overline{\mathbb{B}}$ be the algorithm that runs $\mathbb{B}$ and $\mathbb{A}^*$ in parallel and outputs the output of $\mathbb{B}$ if it is a satisfying assignment and the output of $\mathbb{A}^*$ otherwise. $\overline{\mathbb{B}}$ is obviously a SAT-solver but $(x_s)_{s\in\mathbb{N}}$ is not hard for it. A contradiction to the proposition above. □

For our purposes, a different kind of hard sequence is more useful. In general this new concept is a weaker version of a hard sequence.

Definition 4.13. Let $\mathbb{A}$ be an algorithm solving a search problem R. We say a sequence $(x_y)_{y\in\{0,1\}^*}$ is **probably hard** for $\mathbb{A}$ if

- $y \to x_y$ is ptime computable.
- For every polynomial p and all $s_0, k \in \mathbb{N}$ there is a $s > s_0$ in $\mathbb{N}$ such that the following holds for $\Omega := \{0,1\}^s$:

- For all $y \in \Omega$ there is a $z \in \mathbb{N}$ such that $(x_y, z) \in R$ holds.
- $\underset{y\in\Omega}{Prob}[t_{\mathbb{A}}(x_y) < p(s)] < \frac{1}{k}$ holds.

We can't hope to prove the existence of probably hard sequences for SAT unconditionally as this would imply $P \neq NP$. However, if we assume the existence of hard sequences we can use these to build probably hard sequences.

Proposition 4.5. *If there exists a hard sequence for an algorithm $\mathbb{A}$ solving a search problem R then there exists also a probably hard sequence for $\mathbb{A}$.*

Proof. Let $(x_s)_{s\in\mathbb{N}}$ be a hard sequence for $\mathbb{A}$. Then the sequence $(x^*_y)_{y\in\{0,1\}^*}$ defined by $x^*_y := x_{|y|}$ is a probably hard sequence. Obviously x^*_y is always a positive instance of R because $x_{|y|}$ is by definition . Furthermore, $y \to x^*_y$ is ptime computable because computing $1^|y|$ from y is ptime and then computing x^*_y from this is ptime by definition. Finally, by definition of $(x^*_y)_{y\in\{0,1\}^*}$, we know, for $s \in \mathbb{N}$ and p a polynomial, that $\underset{y\in\{0,1\}^s}{Prob}[t_{\mathbb{A}}(x^*_y) < p(s)]$ is either 0 or 1. Now assume there are p, s_0, k such that for all $s > s_0$ we have $\underset{y\in\{0,1\}^s}{Prob}[t_{\mathbb{A}}(x^*_y) < p(s)] > \frac{1}{k}$. This would imply $\underset{y\in\{0,1\}^s}{Prob}[t_{\mathbb{A}}(x^*_y) < p(s)] = 1$ and consequently $t_{\mathbb{A}}(x_s) < p(s)$ for all $s > s_0$ contradicting the choice of $(x_s)_{s\in\mathbb{N}}$. □

Unfortunately ordinary probably hard sequences don't suffice for our purposes. They need to have also the following additional property.

Definition 4.14. We say a probably hard sequence $(x_y)_{y\in\{0,1\}^*}$ is **invertible** if there is a ptime function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f(x_y) = y$ holds for all $y \in \{0,1\}^*$.

Strongly hard sequences are invertible because we can just extensively try out all possible pre-images. This won't work for probably hard sequences because there are exponentially more possible pre-images. However, under some complexity theoretic assumptions we can produce invertible probably hard sequences from strongly hard sequences.

4.3.2 Paddability and derandomization

In order to make a probably hard sequence invertible we can code the pre-image of a formula directly into the formula. This is possible because SAT is paddable. The following definition of paddability for search problems is a variation of the definition for decision problems from [44].

Definition 4.15. We call a search problem R (polynomially) paddable if there exist two ptime functions $pad(x,y)$ and $d(x)$ such that

- For all $x, y, z \in \{0,1\}^*$ it holds that $(pad(x,y), z)$ is in R if and only if (x,z) is in R.
- For $x, y \in \{0,1\}^*$ we have $d(pad(x,y)) = y$.

All "known" NP-complete problems are paddable (using a similar definition of paddability for decision problems) [7]. For us it is only important that SAT is paddable.

Proposition 4.6. *The search problem SAT is paddable.*

Proof (Sketch). Let $\phi_0 := \top$ and $\phi_1 := \bot$. By definition both are tautologies without propositional variables. Given strings x and y the function pad computes $x \wedge (\top \vee \phi_{y_1} \vee \phi_{y_2} \dots \phi_{y_{|y|}})$ where y_k equals the k-th bit of y if x codes a formula and $\bot \wedge (\top \vee \phi_{y_1} \vee \phi_{y_2} \dots \phi_{y_{|y|}})$ otherwise. The function d is defined in the obvious way. □

However, if we pad the probably hard sequence we constructed in 4.5 with its pre-images, all outputs become unique. Therefore, padding the sequence may destroy the hardness because solving a fraction of formulas out of a set $\{pad(\phi, x) \mid x \in \{0,1\}^n\}$ for some n may be easier than solving the formula ϕ. To eliminate this possibility we need the assumption that randomness does not give additional computational power. This is usually modeled by the assumption that $P = BPP$. Unfortunately in the literature BPP is mainly studied for decision problems but we would need a version of BPP for search problems. Some coverage of BPP search problems can be found in [27]. However, instead of a general assumption about complexity classes we can assume the existence of so called pseudorandom generators to achieve the same effect. We follow [5] in our introduction to (Nisan-Wigderson style) pseudorandom generators. The following definition, a slight variation of the definition in [5], describes a special case of a pseudorandom generator, as we don't need the general theory of pseudorandomness.

Definition 4.16. A 2^n-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ is a **pseudorandom generator** if for some $\delta > 0$ the following holds

- $|G(x)| = 2^{\lfloor \delta |x| \rfloor}$ for all $x \in \{0,1\}^*$
- For all n and every boolean circuit C with at most $(\delta n)^3$ gates the following holds:

$$(\underset{x \in \{0,1\}^{\log(n)}}{Prob} [C(G(x)) = 1] - \underset{x \in \{0,1\}^{\lfloor \delta n \rfloor}}{Prob} [C(x) = 1]) < \frac{1}{10}$$

where $\underset{x \in X}{Prob}[\ldots]$ denotes the probability regarding the uniform distribution over X.

The constants 3 in the size of the circuits can be chosen more or less arbitrary. The existence of such a pseudorandom generator would imply $P = BPP$ as was first shown by Nisan and Wigderson in [37]. On the other hand it is known that pseudorandom generators exist if the complexity class E contains a problem that needs circuits of exponential size to be solved (see [5]). We can apply pseudorandom generators in the following way:

Proposition 4.7. *Let $\mathbb{A}^*$ be the almost optimal SAT-solver. If pseudorandom generators and strongly hard sequences for $\mathbb{A}^*$ exist, an invertible probably hard sequence for $\mathbb{A}^*$ exists.*

Proof. Let $(x_s)_{s \in \mathbb{N}}$ be a strongly hard sequence for $\mathbb{A}^*$. Then we define $x^*_y := pad(x_{|y|}, y)$. By the definition of the padding function this sequence can be generated in ptime. Furthermore, all x^*_y are positive instances of satisfiability. Now assume there are p, s_0, k such that for all $s > s_0$ the following holds

$$\underset{y \in \{0,1\}^s}{Prob} [t_{\mathbb{A}^*}(x^*_y) < p(s)] > \frac{1}{k}$$

Look at the following probabilistic algorithm $\mathbb{C}$ that, given x_s with $s > s_0$, first computes a $s^* > s_0$ such that $x_{s^*} = x_s$ holds by trying out $x_{s_0}, x_{s_0+1}, \ldots$. This takes only polynomial time as the sequence is strongly hard, hence there are only $|x_s|$ many values that have to be checked and each x_i can be computed in polynomial time. We can assume w.l.o.g that we started with x_{s^*} because $x_s = x_{s^*}$ holds. Then the algorithm guesses a string y of length s and then runs $\mathbb{A}^*$ on $pad(x_s, y)$ for $p(s)$ steps. By assumption this solves x_s with probability larger than $\frac{1}{k}$ for $s > s_0$. By running $\mathbb{C}$ on multiple (but constantly many) y we can improve this bound to $\frac{1}{10}$. Therefore, we have constructed an algorithm $\mathbb{B}$ that takes as input an element of the hard sequence x_s and a random string $y \in \{0,1\}^{k(s)}$ for some $k(s)$ polynomially bounded in $|x_s|$ and for all x_s satisfies

$$\underset{y \in \{0,1\}^{k(s)}}{Prob} [\mathbb{B}(x_s, y) \text{ is a satisfying assignment of } x_s] > \frac{1}{10}$$

Furthermore, there exists a polynomial q such that $t_{\mathbb{B}}(x_s, y) < q(|x_s|)$ holds for all x_s and all $y \in \{0,1\}^{k(s)}$.

We can now derandomize this algorithm using a pseudorandom generator $G : \{0,1\}^{\lceil c \log(\frac{1}{\delta} k(s)) \rceil} \to \{0,1\}^{k(s)^c}$ for some δ, where $c \geq 1$ is a constant we pick later. Given x_s we run $\mathbb{B}$ on $(x_s, int(G(y)))$ where $int(G(y))$ denotes the first $k(s)$ bits of $G(y)$ for all $y \in \{0,1\}^{\lceil c\log(\frac{1}{\delta}k(s))\rceil}$. This can be done in polynomial time because in total we have to run $\mathbb{B}$ only $(\frac{1}{\delta}k(s))^c$ many times and $k(s)$ is polynomially bounded in $|x_s|$. We claim that at least one of these runs has to produce a satisfying assignment for x_s for all s bigger than some $s_1 > s_0$, contradicting corollary 4.4. Assume for the sake of a contradiction that there is an infinite sequence of $(x_{s_i})_{i\in\mathbb{N}}$ such that

$$\underset{y\in\{0,1\}^{\lceil c\log(\frac{1}{\delta}k(s_i))\rceil}}{Prob}[\mathbb{B}(x_{s_i}, int(G(y))) \text{ is a satisfying assignment of } x_{s_i}] = 0$$

holds for all $i \in \mathbb{N}$. Then, for every i, we construct a circuit $C_i : \{0,1\}^{k(s_i)^c} \to \{0,1\}$

$$C_i(y) := \begin{cases} 1 & \text{if } \mathbb{B}(x_{s_i}, int(y)) \text{ is a satisfiying assigment of } x_{s_i} \\ 0 & \text{otherwise} \end{cases}$$

This can be done in a way such that C_i has $k(s_i)^c + k(s_i)^d$ gates for some constant d depending on $\mathbb{B}$ but not on c, by constructing a circuit C_i with $k(s_i)^c$ input gates that ignores all but the first $k(s_i)$ input gates. Hence, we can pick a c such that $c > d$ holds. Then, C_i has less than $(k(s_i))^{3c}$ gates. However, by the definition of $\mathbb{B}$ we know for all C_i

$$(\underset{y\in\{0,1\}^{\lceil c\log(\frac{1}{\delta}k(s))\rceil}}{Prob}[C_i(G(y)) = 1] - \underset{y\in\{0,1\}^{k(s_i)}}{Prob}[C_i(y) = 1]) \geq \frac{1}{10}$$

contradicting the assumption that G is a pseudorandom generator. □

4.3.3 Pseudo proof systems that err everywhere

We can now prove the main result of this section. Invertible probably hard sequences exist if and only if pseudo proof systems that err everywhere (see def 4.7) do. To enhance the readability we will split this result in two theorems, one for each direction.

Theorem 4.8. *If there is an invertible probably hard sequence for the almost optimal SAT-solver $\mathbb{A}^*$ then there is $n \in M \setminus \mathbb{N}$ such that $K(F^n_{PV})$ contains a pseudo proof system that errs everywhere.*

Proof. Let $(x_y)_{y\in\{0,1\}^*}$ be a invertible probably hard sequence for $\mathbb{A}^*$ and let f be the function defined by $f(y) := x_y$ which is ptime by definition. Consider the type $p(l)$ that contains for every ptime function g and every $k \in \mathbb{N}$ a formula $\phi^k_g(l)$ stating that '$g(y)$ is a satisfying assignment for $f(y)$ for less than a $\frac{1}{k}$ fraction of strings y of length l and $f(y)$ is satisfiable for all strings y of length l'. The first part of this formula looks like this.

$$\psi_g^k(l) = \underset{|y|=l}{Prob}[Sat(f(y), g(y))] < \frac{1}{k}$$

To formalize the second part we use the satisfiability relation $Sat(x, y)$ from definition 4.3

$$\xi(l) = \forall y \in \{0,1\}^l \exists z \; Sat(f(y), z)$$

Finally we can formalize ϕ_g^k as $\phi_g^k(l) := \psi_g^k(l) \wedge \xi(l)$ and $p(l)$ as

$$p(l) := \{\phi_g^k(l) \mid k \in \mathbb{N}, g \in L_{all}, g \text{ is ptime}\}$$

We claim that $p(l)$ is a M-type (over the empty set as L_{all} contains symbols for all k and g). Otherwise there would be a finite set of ptime functions $g_1, g_2, \ldots g_m$ and a finite set of natural numbers $k_1, k_2, \ldots k_m$ such that no $l \in M$ satisfies $\phi_{g_i}^{k_i}$ for all $i < m$. Now let g^* be the ptime function one gets by running $g_1, g_2, \ldots g_m$ in parallel and $k^* = max(k_1, k_2, \ldots k_m)$. Then g^* finds a satisfying assignment on every input on which at least one g_i finds one. Now we know that for every l^* at least one formula $\phi_{g_i}^{k_i}(l^*)$ is wrong. This means for every l^* either not all $\{f(y) \mid y \in \{0,1\}^{l^*}\}$ are satisfiable or g^* finds for at least a $\frac{1}{k^*}$-fraction of $y \in \{0,1\}^{l^*}$ a satisfying assignment for $f(y)$. We can write this statement as

$$\forall l^*(\xi(l^*) \rightarrow \underset{|y|=l^*}{Prob}[Sat(f(y), g^*(y))] \geq \frac{1}{k^*})$$

But then look at the algorithm $\mathbb{B}$ that, given x_y, computes y andthen computes $g^*(y)$. This is obviously a polynomial time algorithm because $(x_y)_{y\in\{0,1\}^*}$ is invertible. For all l^* such that $\xi(l^*)$ holds we have

$$\underset{|y|=l^*}{Prob}[Sat(x_y, \mathbb{B}(x_y))] \geq \frac{1}{k^*} \quad (1)$$

However, this is not possible because $\mathbb{A}^*$ is bounded by some polynomial $q(x)$ on every input where $\mathbb{B}(x_y)$ is a satisfying assignment of x_y by the argument used in the proof of corollary 4.4. Hence, (1) would imply that we can not find a l^* such that $\xi(l^*)$ holds and $\mathbb{A}^*$ is bounded by $q(x)$ on less than a $\frac{1}{k^*}$-fraction of x_y for $y \in \{0,1\}^{l^*}$. However, this contradicts the assumption that $(x_y)_{y\in\{0,1\}^*}$ is probably hard. Therefore, $p(l)$ is a type.

M is $\aleph_1$-saturated therefore it realizes the type $p(l)$ trough some $n \in M$. Now consider the model $K(F_{PV}^n)$. Now let $\overline{f}$ be the function that maps y to the negation of $f(y)$. Then $\overline{f}$ is a pseudo proof system that errs everywhere.

First it is clear that it errs everywhere because $f(y)$ is a positive instance of satisfiability in M which means $\overline{f}(y)$ can not be a tautology in M. Secondly

$$\underset{\Omega}{Prob}[Sat(f(k), g(k))] < \epsilon$$

for every standard ϵ and every ptime function g by the choice of n which implies

$$\bigwedge_g [\![|g| \leq |f| \rightarrow \mathrm{Sat}(\overline{f}, g)]\!] = 1_{\mathcal{B}}$$

Therefore, $\overline{f}$ is a pseudo proof system. □

Theorem 4.9. *If there is a model $K(F_{PV}^n)$ for some nonstandard n with a pseudo proof system that errs everywhere then there is a invertible probably hard sequence for the almost optimal SAT-solver $\mathbb{A}^*$.*

Proof. Let $K(F_{PV}^n)$ be a model with a pseudo proof system that errs everywhere. Furthermore, let $\alpha_f \in F_{PV}$ be a pseudo proof system that errs everywhere, where f is a total ptime function such that $f(x) = \alpha_f(x)$ holds for all $x \in \{0,1\}^n$. Then in M

$$\underset{y \in \{0,1\}^n}{Prob}[|g(y)| < |f(y)| \rightarrow g(y) \text{ is a satisfying assigment for } f(y)] > 1 - \frac{1}{k}$$

holds for all ptime functions g and all $k \in \mathbb{N}$ by definition. This implies obviously

$$\underset{y \in \{0,1\}^n}{Prob}[g(y) \text{ is a satisfying assigment for } \neg f(y)] < \frac{1}{k} = \phi_k^g(n)$$

Furthermore, we write as in the proof above

$$\xi(n) := \forall y \in \{0,1\}^n \exists z \; Sat(f(y), z)$$

Then we know for every $l \in \mathbb{N}$ and all g and k the following holds in M

$$\exists m > l \; (\phi_k^g(m) \wedge \xi(m))$$

As this is a sentence in M without parameters this statement is also true in $\mathbb{N}$. Now we claim the sequence defined by $x_y := pad(\neg f(y), y)$ is a invertible probably hard sequence. It is obviously invertible. Furthermore, x_y can be generated in ptime as f and pad are ptime and our encoding is reasonable. Now assume there would be a polynomial p and $s_0, k \in \mathbb{N}$ such that for all $s > s_0$

$$(\underset{y\in\{0,1\}^s}{Prob}[t_{\mathbb{A}^*}(x_y) < p(s)] > \frac{1}{k}) \wedge \xi(s)$$

Then the ptime function g^* that on input y computes $\mathbb{A}^*$ on x_y for $p(|y|)$ steps would satisfy

$$\underset{y\in\{0,1\}^s}{Prob}[g^*(y) \text{ is a satisfying assigment for } \neg f(y)] > \frac{1}{k}$$

for all $s > s_0$. But this is a contradiction.

□

As a corollary we can answer one of the questions posed above.

Corollary 4.10. *If invertible probably hard sequences for the optimal SAT-solver exist there is a $n \in M \setminus \mathbb{N}$ such that no proof system P approximately p-simulates all pseudo proof systems in $K(F_{PV}^n)$.*

Proof. A pseudo proof system that errs everywhere can obviously not be approximately p-simulated by a propositional proof system. □

This yields a $Th_\forall(\mathbb{N})$-model $K(F_{PV}^n)$ where we have a single element α such that for all proof systems P we get $[\![Taut(\alpha) \wedge \neg P \vdash \alpha]\!] = 1_{\mathcal{B}}$ i.e. there is a unique counter example for the completeness of all proof systems.

4.4 Global pseudo proof systems

We want to end this chapter with a proposal for a further line of investigation regarding pseudo proof systems. We want to lift the 'local' definitions of this chapter to a more global context. This seems natural as pseudo proof systems and approximate p-simulations are restrictions of functions $M \to M$. Therefore, we can ask about properties of functions such that the restriction to $\{0,1\}^n$ is a pseudo proof system or a approximate p-simulation for every nonstandard $n \in M$.

Definition 4.17. We say a ptime function P of M is a **global pseudo proof system** if for all $n \in M \setminus \mathbb{N}$ its representation α_P is a pseudo proof system in $K(F_{PV}^n)$.

Let P and Q be global pseudo proof systems in M. We say P **globally approximately p-simulates** Q if there is a ptime function f such that, for all $n \in M \setminus \mathbb{N}$, the formula '$P(\alpha_f) = \alpha'_Q$ is $K(F_{PV}^n)$-valid .

It is straight forward to prove that these definitions are proper extensions of the concept of a proof system and p-simulation respectively.

Example 4.2. Let P be a proof system in M. We define $\overline{P}$ in the following way

$$\overline{P}(x) := \begin{cases} P(x) & \text{if } x \neq 0^k \text{ for some } k \in M \\ \perp & \text{if } x = 0^k \text{ for some } k \in M \end{cases}$$

Then $\overline{P}$ is obviously a global pseudo proof system but not a proof system.

Now let Q be a truth-table proof system formalized in a way such that 0^k is not the unique proof of a tautology for any $k \in M$. Then, Q needs proofs of exponential size for every tautology. Now pick an arbitrary infinite sequence of tautologies $(\phi_k)_{k \in M}$ constructible in ptime such that $|\phi_s| > s$ holds. Then we define a proof system $\overline{Q}$ in the following way

$$\overline{Q}(x) := \begin{cases} P(x) & \text{if } x \neq 0^k \text{ for some } k \in M \\ \phi_s & \text{if } x = 0^s \end{cases}$$

Then Q does not p-simulate $\overline{Q}$ as 0^s is polynomial in the size of ϕ_s and Q has only exponential size proofs of ϕ_s. However, Q globally approximately p-simulates $\overline{Q}$ by the identity function.

This shows that we haven't made trivial definitions. Therefore, we can now study properties of these definitions. Unfortunately global approximate p-simulation of global pseudo proof systems does not satisfy the most basic property of an order, it is not transitive.

Example 4.3. Let P be a propositional proof system. Let ϕ be a arbitrary standard tautology. We define P^* as $P^*(x) := P(x)$ if $P(x) \neq \phi$ and $P^*(x) = \psi$ else for some tautology $\psi \neq \phi$. It is easy to see that we can do this in a way that P^* is ptime. Now we define Q as $Q(x) := P^*(x)$ if x is not of the form 0^s for some $s \in M$ and $Q(0^s) = \phi$. As all these functions put out only tautologies they are all (global) pseudo proof systems and it is easy to see that P^* approximately p-simulates Q via the identity as they only differ on one input from $\{0,1\}^k$ for every (nonstandard) k. Now we define another global pseudo proof system $R(x) = \phi$ for all $x \in \{0,1\}^*$. Then P^* does not globally approximately p-simulates R as $P^*(x) \neq \phi$. However, Q globally approximately p-simulates R by the constant function $f(x) = 0$.

We propose a variation of the notion of an approximate p-simulation that retains transitivity. To do so, we have to ask our self how global approximate p-simulations look as functions of $\mathbb{N}$.

Proposition 4.11. *Let f, P and Q be L_{all} function symbols. P globally approximately p-simulates Q by f in M, if and only if the following holds in $\mathbb{N}$: For all $\epsilon > 0$ there is a $n_0 \in \mathbb{N}$ such that for all $n > n_0$ it holds that*

$$\underset{x \in \{0,1\}^n}{Prob}[P(f(x)) = Q(x)] > 1 - \epsilon$$

Proof. Assume P globally approximately p-simulates Q by f in M. Then by lemma 2.12 we have

$$\underset{x \in \{0,1\}^n}{Prob}[P(f(x)) = Q(x)] > 1 - \epsilon \quad (*)$$

for all standard ϵ and all nonstandard n. Fix any standard $\epsilon > 0$ and assume that there is no n_0 with the desired property. Then by overspill (proposition 1.4) there is some nonstandard n such that (*) is false. Contradiction.

Now assume in $\mathbb{N}$ there is, for all $\epsilon > 0$, a $n_0 \in \mathbb{N}$ such that (*) holds for all $n > n_0$. Then for all standard $\epsilon > 0$ we know that (*) holds in M for all $n > n_0$ with $n \in M$. As all n_0 are standard, this implies that (*) holds for all standard ϵ for all nonstandard n. Because '$P(f(x)) = Q(x)'$ is an atomic formula this implies $[\![P(\alpha_f) = \alpha_Q]\!] = 1_{\mathcal{B}}$ for $K(F^n_{PV})$ for all $n \in M \setminus \mathbb{N}$. Hence, P globally approximately simulates Q by f in M. □

If we were to change in this representation the probability with respect to the uniform distribution to probability with respect to every polynomial time sampleable distribution we would trivially get a transitive notion of approximate p-simulation. However, as this approach can no longer be represented easily in the context of the models $K(F^n_{PV})$, we have strayed far away from the topic of this thesis. Therefore, further discussion of these ideas is left for future work.

Conclusion

We may assume that Jan Krajíček's intend in writing [33] was the establishment of new proof complexity lower bounds. He wasn't able to achieve this goal. Nevertheless, the forcing method presented in this book provides new research directions that "are highly intriguing as an new approach for attacking fundamental problems in proof complexity" [11]. Therefore "the first parts of the book should be interesting to anyone working in model theoretic constructions for non-standard models of arithmetic" [11].

Moreover, the forcing method provides a uniform framework for proving many different (independence) results. Therefore, it can be advantageous for the student of bounded arithmetic to study this method in order to approach many different results using the same method.

We believe, therefore, that further research on this method would be a worthwhile undertaking. We hope to have made, with this thesis, a small contribution to this project, by making the forcing method of Krajíček accessible to a wider audience. Especially, we hope that the first three chapters enable an attentive reader to understand and apply the basics of the forcing method, even without prior knowledge in bounded arithmetic.

Furthermore, we hope that we have shown, especially in the last chapter, that the forcing framework may be useful not just for independence proofs. In particular the topic of pseudo proof systems provides, in our opinion, many possible directions for further research. Many of the open problems about propositional proof systems can be reformulated as problems about pseudo proof systems, giving new, probably easier, versions of these long open questions. Moreover, removed from the forcing method, it may be fruitful to further develop the concept of function that is nearly a proof system, i.e a "pseudo" proof system, in the standard model. A possible starting point could be the definition proposed at the very end of chapter 4.

Finally, apart from the topics related to Jan Krajíček's book, it could be interesting to have a closer look at probably hard sequences, studying their relation to other forms of hard sequences and consequences of their existence.

Abstract

Diese Arbeit behandelt eine Forcing Methode, die Jan Krajíček in dem Buch ‘Forcing with random variables and proof complexity’ ([33]) 2010 vorgestellt hat und richtet sich an Leser mit grundlegenden Kenntnissen in Logik und Komplexitätstheorie. Die Forcing Methode erlaubt es Modelle für verschiedene Theorien aus dem Bereich der ‘Bounded Arithemtic’ zu konstruieren.

Es werden die mathematischen Grundlagen, die zum Verständis der Methode notwendig sind, diskutiert, insoweit sie über den üblichen Umfang einer Einfürungsvorlesung in Logik und Komplexitätstheorie hinausgehen. Danach wird die Forcing Methode allgemein vorgestellt und anschließend an einem Beispiel vorgeführt.

Des weiteren werden so genannte ‘pseudo proof systems’ behandelt, die ebenfalls von Jan Krajíček in [33] vorgestellt wurden. Hierbei handelt es sich um Funktionen, die sich, im Kontext der mit der Forcing Methode konstruierten Modelle, wie Beweissysteme verhalten, aber, im Allgemeinen, keine Beweissysteme sind. Diese ‘pseudo proof systems’ werden eingeführt und ein neues Ergebnis wird vorgestellt. Dazu werden ‘pseudo proof systems that err everywhere’ definiert. Hierbei handelt es sich um ‘pseudo proof systems’ die ausschließlich Sätze beweisen, die keine Tautologien sind. Anschließend wird bewiesen, dass solche Funktionen genau dann existieren, wenn gewisse harte Sequenzen, ‘invertible probably hard sequences’ existieren. Diese werden in dieser Arbeit neu eingeführt. Ihre Existenz lässt sich unter bekannten komplexitätstheorethischen Annahmen beweisen. Abschließend wird kurz die Möglichkeit diskutiert das Konzept eine ‘pseudo proof systems’ auch über den Kontext der Forcing Methode hinaus fruchtbar zu machen.

Bibliography

[1] Scott Aaronson. Why philosophers should care about computational complexity. In *Computability: Gödel, Turing, Church, and beyond*, pages 261–329. MIT Press, 2013.

[2] Miklós Ajtai. The complexity of the pigeonhole principle. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 346–355. IEEE, 1988.

[3] Miklos Ajtai. *Parity and the pigeonhole principle.* Springer, 1990.

[4] Miklos Ajtai. The independence of the modulo p counting principles. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 402–411. ACM, 1994.

[5] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[6] Albert Atserias and Moritz Müller. Partially definable forcing and bounded arithmetic. *Archive for Mathematical Logic*, (54(1-2)):1–33, 2015.

[7] Leonard Berman and Juris Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[8] Kostas Bimpikis and Ragesh Jaiswal. Modern factoring algorithms. http://www.cs.columbia.edu/ rjaiswal/factoring-survey.pdf, 2005.

[9] Dan Boneh and Ramaratham Venkatesan. Breaking RSA may not be equivalent to factoring. In *Advances in Cryptology-Eurocrypt'98*, pages 59–71. Springer, 1998.

[10] George Boole. *The mathematical analysis of logic.* Philosophical Library, 1847.

[11] Sam Buss. Krajíček Jan. Forcing with random variables and proof complexity. London Mathematical Society Lecture Note Series, vol. 232. Cambridge University Press, 2011, xvi+ 247 pp. *Bulletin of Symbolic Logic*, 18(04):576–578, 2012.

[12] Samuel R Buss. *Bounded arithmetic*, volume 86. Bibliopolis Naples, 1986.

[13] Samuel R Buss. Towards NP-P via proof complexity and search. *Annals of Pure and Applied Logic*, 163(7):906–917, 2012.

[14] Chen Chung Chang and H Jerome Keisler. *Model theory.* Elsevier, 1990.

[15] Yijia Chen, Jörg Flum, and Moritz Müller. Hard instances of algorithms and proof systems. *ACM Transactions on Computation Theory (TOCT)*, 6(2):7, 2014.

[16] Paul J Cohen. The independence of the continuum hypothesis. *Proceedings of the National Academy of Sciences*, 50(6):1143–1148, 1963.

[17] Stephen A Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of seventh annual ACM symposium on Theory of computing*, pages 83–97. ACM, 1975.

[18] Stephen A Cook and Robert A Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(01):36–50, 1979.

[19] Whitfield Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76(5):560–577, 1988.

[20] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[21] G Lejeune Dirichlet. Recherches sur les formes quadratiques à coefficients et à indéterminées complexes. Première partie. *Journal für die reine und angewandte Mathematik*, 24:291–371, 1842.

[22] G Lejeune Dirichlet. Verallgemeinerung eines Satzes aus der Lehre von den Kettenbrüchen nebst einigen Anwendungen auf die Theorie der Zahlen. *SB Preuss. Akad. Wiss*, pages 93–95, 1842.

[23] Herbert B Enderton. *A mathematical introduction to logic.* Academic press, 2001.

[24] Haim Gaifman. Concerning measures on boolean algebras. *Pacific J. Math*, 14:61–73, 1964.

[25] Anthony A Gioia. *The theory of numbers: an introduction.* Courier Corporation, 2001.

[26] Kurt Gödel. *Über die Vollständigkeit des Logikkalküls.* PhD thesis, University of Vienna, 1929.

[27] Oded Goldreich. In a world of P=BPP. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 191–232, 2011.

[28] Leila Haaparanta. *The development of modern logic.* Oxford University Press, 2009.

[29] Wilfrid Hodges. *Model theory*, volume 42. Cambridge University Press Cambridge, 1993.

[30] Jeffrey Hoffstein, Jill Catherine Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography.* Springer, 2008.

[31] Thomas Jech. *Set theory.* Springer Science & Business Media, 2013.

[32] Jan Krajíček. *Bounded arithmetic, propositional logic and complexity theory.* Cambridge University Press, 1995.

[33] Jan Krajíček. *Forcing with random variables and proof complexity*, volume 382. Cambridge University Press, 2010.

[34] Jan Krajíček and Pavel Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.

[35] Kenneth Kunen. *Set theory an introduction to independence proofs*, volume 102. Elsevier, 2014.

[36] Leonid A Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[37] Noam Nisan and Avi Wigderson. Hardness vs. randomness. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 2–11. IEEE, 1988.

[38] Rohit Parikh. Existence and feasibility in arithmetic. *The Journal of Symbolic Logic*, 36(03):494–508, 1971.

[39] Jeff Paris and Alex Wilkie. Counting problems in bounded arithmetic. In *Methods in mathematical logic*, pages 317–340. Springer, 1985.

[40] Alexander A Razborov. Proof complexity of pigeonhole principles. In *Developments in Language Theory*, pages 100–116. Springer, 2001.

[41] Søren Riis. *Independence in bounded arithmetic.* PhD thesis, University of Oxford, 1994.

[42] Benoît Rittaud and Albrecht Heeffer. The pigeonhole principle, two centuries before dirichlet. *The Mathematical Intelligencer*, 36(2):27–29, 2014.

[43] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[44] Uwe Schöning. *Complexity and Structure.* Lecture Notes in Computer Science 211. Springer-Verlag Berlin Heidelberg, 1 edition, 1986.

[45] Dana Scott. A proof of the independence of the continuum hypothesis. *Theory of Computing Systems*, 1(2):89–111, 1967.

[46] Joel A Tropp. *Infinitesimals: History & Application.* PhD thesis, The University of Texas at Austin, 2004.

[47] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.